SBC FTP Server and SBC Flash file system

Document 26-855; Edition ENG08 | 2016-12-12

0

0

0.1 Document History

Date	Version	Changes	Remarks
2007-07-09	EN01	-	Initial Edition
2008-05-20	EN02	Ch 3.3.3 Ch C.2	Configuration file location Web-Address
2008-06-05	EN03	Ch 3.3.2 Ch 3.3.4	GroupID 0xFF --> 0x00 GroupID 0xFF --> 0x00
2008-09-16 2008-11-13	EN04	Ch 1 Ch B.2.3.1 Ch B.2.3.1 Ch B.3.15 Ch 2.3	FW-version of the PCD2.M5xx0 Table replaced Table replaced Parameters modified Renamed PCD7.R651 --> PCD7.R561
2009-01-09	EN05	Ch 3.3.4	small corrections in code example
2011-07-15	EN06	Appendix B	Doubleword instead of byte, in table position “Seek.Pos” page B-5
2013-10-23	EN07	-	New logo and new company name
2016-12-12	ENG08	-	New phone number

0.2 About this manual

See the section in the appendix in relation to some of the terms, abbreviations and the references used in this manual.

0.3 Brands and trademarks

Saia PCD® and Saia PG5®
are registered trademarks of Saia-Burgess Controls AG.

Windows® and Microsoft® are registered trademarks of Microsoft Corporation.

Technical modifications are based on the current state-of-the-art technology.

Published in Switzerland

1 Introduction

This document is intended to present the features associated with the new firmware modules FTP-Server and_flash file system, as well as to provide information about the_flash storage media hardware modules.

This document is valid for Saia PCD® Classic users.

The FTP-Server and the SBC File System (SFS) have been implemented in the PCD3 firmware version ≥020.

The asynchronous System Functions for the SFS have been implemented in the PCD3 firmware version ≥039.

On the PCD2.M5xx0, all functionalities described in this manual are available from the first official version (1.08.19).

Saia PCD® systems other than PCD3 and PCD2.M5xx0 do not support the SFS and the FTP server.

1.1 TCP/IP Setup

In order to have the FTP-Server, HTTP-Direct and/or the HTTP-Server running, it is necessary to give an Internet address.

This is done by giving an IP address, e.g. 192.168.1.12 and a subset-mask, e.g. 255.255.255.0 in the hardware setting window of a Saia PG5® project / defined CPU. After the download of this configuration the Saia PCD® need to be powered up for that the servers are started.

Without the IP-configuration, any accesses to FTP-Server or access to Web Server over http-direct will not work.

2 Flash behaviour and flash devices

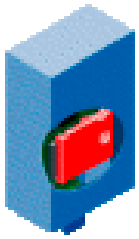
Flash memory in general is non-volatile memory that can be electrically erased and reprogrammed. The flash memory of the modules dedicated to file system can be written in run time of the PCD3 and the PCD2.M5xxx. As these devices to feature a file system, “common” files as they are used on PC systems can be stored, read and written on the device. Typical areas of usage are:

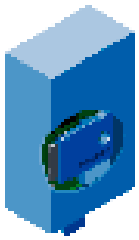
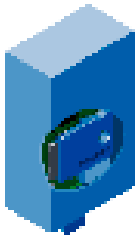
- Storing files that can be accessed over the Web Server (the IMaster*.jar and other files for Web Editor projects, pictures as well as files like manuals in pdf format, etc.)
- Storing log files written by the Saia PCD® in e.g. csv format (Comma delimited files)
- Storing configuration files for the Saia PCD® or the user program

Flash memory devices dedicated to file system can be accessed by the following methods:

- By the Saia PCD® user program (read/write/format and compress access)
- By the internal FTP server of the Saia PCD® (read/write access)
- By the internal Web Server of the Saia PCD® (read access, for files in the WEBPAGES folder, file system management (reading status, compression and formatting by using CGI calls), format and compress access)
- SD cards that are used in the PCD3.R600 module can be read an written using a PC equipped with an SD card reader (with the software SBC SD File system Explorer)

2.1 Available flash devices dedicated to file system

Memory modules for PCD3.Mxxxx CPUs			
Module	Picture	Description	For System
PCD7.R500		The PCD7.R500 is a flash memory module with 1 MByte for PCD3.M5xxx and PCD3.M6xxx as backup of the user program. This module is to be inserted into slot M1 or M2 on the PCD3.	PCD3.M5xxx PCD3.M6xxx
PCD3.R500		Flash memory module with 1 MByte for PCD3.M3xxx as backup of the user program. The PCD3.R500 is to be inserted into I/O slot 0..3 of the PCD3.M3xxx. It contains a PCD7.R500 memory module.	PCD3.M3xxx

Memory modules for PCD3.Mxxxx CPUs			
Module	Picture	Description	For System
PCD7.R550M04		Flash memory module with 4 MByte file system for a PCD3.M5xxx and PCD3.M6xxx. On this module files can be stored for e.g. the Web Server of the PCD3. The files stored on a PCD7.R550M04 are accessible over the FTP- or HTTP direct server of the PCD3. The Saia PCD® can also write PC-readable files such as *.csv files directly on the file system of this module.	PCD3.M5xxx PCD3.M6xxx
PCD3.R550M04		Flash memory module with 4 MByte file system for a PCD3.M3xxx to be inserted into I/O slot 0..3. Files for e.g. the Web Server of the PCD3.M3xxx can be stored on this module. The PCD3.R550M04 contains a PCD7.R550M04. The files stored on this module are accessible over the FTP- or HTTP direct server of the PCD3. The Saia PCD® can also write PC-readable files such as *.csv files directly on the file system of this module.	PCD3.M3xxx
PCD7.R551M04		Flash memory module with 3 MByte file system and 1 MByte for a user program backup on a PCD3.M5xxx and PCD3.M6xxx. The files stored on file system on a PCD7.R551M04 are accessible over the FTP- or Web server of the PCD3. The Saia PCD® can also write PC-readable files such as *.csv files directly on the file system of this module.	PCD3.M5xxx PCD3.M6xxx
PCD3.R551M04		Flash memory module with 3 MByte file system and 1 MByte for a user program backup on a PCD3.M3xxx. The PCD3.R551M04 contains a PCD7.R551M04 and is to be plugged in one of the I/O slots 0..3.	PCD3.M3xxx
PCD7.R560		Flash memory module with BACnet firmware for PCD3.M5540. The PCD7.R560 holds the firmware extension for BACnet as well as the configuration files for the BACnet application. This module is to be plugged in onto slot M1 or M2 of the PCD2.M5540.	PCD3.M5540
PCD3.R560		Flash memory module with BACnet firmware for PCD3.M3330 or PCD3.M3120. The PCD3.R560 contains a PCD7.R560 that holds the firmare extension for BACnet as well as the configuration files for the BACnet application. This module can be insterted in one of the I/O slots 0..3.	PCD3.M3330 PCD3.M3120

Memory modules for PCD3.Mxxxx CPUs			
Module	Picture	Description	For System
PCD7.R561		Flash memory module with BACnet firmware for PCD3.M5540, 2 MBytes file system and 1 MByte for the backup of the user program. The PCD7.R561 is to be plugged in onto slot M1 or M2 of the PCD2.M5540.	PCD3.M5540
PCD3.R561		Flash memory module with BACnet firmware for PCD3.M3330 or PCD3.M3120, 2 MBytes file system and 1 MByte for user program backup. The PCD3.R561 can be insterted in one of the I/O slots 0..3.	PCD3.M3330 PCD3.M3120
PCD3.R600		Basic module for SD flash memory card holding a file system and one MByte for user program backup. The module is to be plugged in one of the I/O slots 0..3 of any PCD3 CPU. The files stored on a PCD3.R600 are accessible for the PCD3 Web Server or the FTP server. It is possible writing PC-readable files from the PCD3 user program. A PCD7.R-SDxxx card is to be inserted into this module.	PCD3.Mxxxx
PCD7.R-SD256 PCD7.R-SD512		SBC SD flash memory card with 256 MBytes (PCD7.R-SD256) or 512 MBytes (PCD7.R-SD512) file system, to be used in a PCD3.R600. These SD flash card can be read using a card reader and a special software on the PC.	PCD3.R600
PCD3.R010		Battery module for PCD3.M3xxx, plug-in onto I/O Slot #3.	PCD3.M3xxx

Note that the maximal number of flash devices with file system or BACnet option is 4 per PCD3 CPU.

2.2 Specific flash behaviour

2.2.1 Internal structure

Flash devices dedicated to file system are organized into pages (256 / 512 bytes [1]), blocks (512 Bytes…8 KBytes [1]) and sectors (64 KB). A page is a unit of write, a block is a unit of file and a sector is a unit of erase. This means that the smallest part of the flash memory that can be written is a page of 256 bytes while the smallest area of a flash device that can be erased at once is 64 KBytes.

If a file is created, at least one block (e.g. 2 KBytes) is allocated to that file. Even if the file is smaller than this, the place taken on the device in this case is 2 KB. When the 2 KB are filled with file data, a new block will be taken and the size taken on the device will be 4 KB.

On each module dedicated to flash file system some space is reserved for file system internal data (one block of 64 Kbytes is reserved and some blocks are used for file system internal data). This means that a bit less than the specified flash size can be used by the user.

2.2.2 Deleting and erasing data on flash devices

Unlike to information stored on e.g. RAM (Random Access Memory), it is not possible modifying information just by addressing and changing the value of the according bits on flash memory. Before a part of flash memory can be re-written after e.g. the deletion of data, the according part needs to be erased. By erasing memory, all bits of this part are to be set to 1.

If a file is deleted on a flash device, the according area on the device is marked as “freed” but not immediately erased (because the smallest area that can be erased is a sector and this sector can contain many files and not only the one to be deleted). As result, it is not immediately possible re-using the memory that has been occupied by the deleted file. The memory first needs to be compressed (→ erasing memory which is marked as “freed”).

2.2.3 Compressing a flash device

The compression of a flash device is a complex task that copies all used data of a sector to a reserved sector and then erases the sector to be compressed. After the erase process (which takes about 2 seconds) the used data is copied back to its original sector (and the reserved sector is erased as preparation for the next cycle). After the compression, there is no more “freed” memory on the compressed sector and therefore all unused memory can be written.

By default the compression of the device is triggered automatically (based on the ratio of used, freed and available memory on the device). During the compression of a flash device, this device can not be accessed and is marked as “busy”. This can lead to the fact that e.g. a download of a file using FTP fails because the Saia PCD® starts compressing in the middle of the download.

In order to avoid this kind of trouble it is recommended compressing the device by either the user program or the CGI calls on the Saia PCD® Web Server.

1) The size of the pages and blocks is depending on the device; SD cards placed in a PCD3.R600 generally do have bigger pages and blocks.

Note that during the compression algorithm the Saia PCD® must not be switched off. Switching of the Saia PCD® while compressing can lead to data loss or a corruption of the file system.

In order to avoid excessive compress activity (leading to a fast wear-out of the flash and to frequent state “busy” due to compression) the following maximal memory usage of flash devices is recommended:

- 50% of the size of the device in case data is written by the user program (e.g. log files) [1])
- 80% in case files are stored statically only (for Web Server access only)

The compression of a flash device can take up to several minutes depending on the amount of sectors that are to be compressed (about 4 seconds per sector on a PCD7.R55xM04). During this time the flash device is marked as “busy”.

A compression algorithm does always compress all sectors that contain “freed” memory of the device.

The maximum number of write cycles of a flash memory cell is limited to about 100’000 (due to the wear-out of the flash memory). Since not all cells are concerned by a compress algorithm, the actual amount of write cycles possible is higher than 100’000.

2.2.4 SBC File system specific features

Since the commonly used formats of flash file systems (FAT, NTFS etc) are not well suitable for the use on a PLC, the flash modules supporting file system on Saia PCD® controllers are formatted with the SBC File System (SFS). One of the advantages of the SFS is, that it is possible writing smaller pages and blocks (and therefore writing faster and using the space on the flash more economically).

All modules supporting file system are sold formatted with the SBC File System.

The following features do apply to all SBC flash modules supporting file system and need to be considered for its use:

- The SBC File system features a maximum number of files of 900 per file system (flash module). Each directory that is created on a file system (flash module) does reduce the maximum number of files by 20.
- Each file and directory does have a GroupID property. A file can only be accessed if the AccessGroup of the user (e.g. FTP user of the Saia PG5® FBox that accesses the file) involves the GroupID.
  A file can only be created within a folder if the AccessGroup parameter of the user creating the file involves the GroupID of the folder. Please refer to the chapter “FTP user configuration” for more information.
- The maximum file name length is 23 characters including the file extension (e.g. .csv). A file or folder name must not contain a space character or a double point.
  It is not recommended using special characters (umlauts etc.)
- The maximum absolute file name length (file name including path, e.g. M2_

1) If more than 50% of the file system space is used on a flash device on which files are written frequently, it is strongly recommended that the compression tasks are managed (triggered) by the user program.

FLASH:/FOLDER/FILE.txt) is 63 characters

- There is no “creation time” or “last modification time” information for flash devices, files or folders.
- File and folder names are always upper case. If lower case names are given, they are converted to upper case by the Saia PCD®.

2.3 PCD7.R5xx devices

The devices PCD7.R550M04 and PCD7.R551M04 are hot pluggable. This means that they can be inserted at any time during the Saia PCD® operations. Immediately after its insertion, the file system is built and eventually, a compression algorithm is automatically executed if compression criteria are met. This implies that at insertion time, the device may be visible to users, but it is possible that the device is marked as busy.

Also the devices PCD7.R56x are hot pluggable, but only the file system but no BACnet will be available without a power off/on cycle.

For more information regarding the PCD7.R500, please refer to the PCD3 hardware manual.

Even if the PCD7.R550M04 is electrically hot un-pluggable, the removal of a device shall be carefully done. This device has no LED providing information if the device is currently accessed or not.

Two cases shall be carefully studied when a flash device is removed:

- Some write operations are currently going on. These operations can be performed either by the user program or by FTP. Before removing the device, put the Saia PCD® in STOP if the user program is writing on a flash device and make sure that no FTP connection is currently on-going.
- The compression algorithm is currently going on. This operation is either triggered by the user program (directly triggered or indirectly through erasing of files) or indirectly by FTP after erasing files. These actions are difficult to detect by external users.

For these reasons, and even if the removal of a flash device is not strictly forbidden, it is highly NOT recommended to remove a flash device while the system is running.

The inserted flash devices will have the following names:

- M1_FLASH when a PCD7.R55xM04 or a PCD7.R561 is inserted in M1 slot of the PCD3.M5xxx or PCD2.M5xxx
- M2_FLASH when a PCD7.R55xM04 or a PCD7.R561 is inserted in M2 slot of the PCD3.M5xxx or PCD2.M5xxx

An absolute path to access a file will look like:

M2_FLASH:/MYFOLDER/MyFile.txt

The file names and directory names are case insensitive (They will be converted to upper case by the Saia PCD®).

2.4 PCD3.R5xx devices

The modules PCD3.R5xx are PCD3 I/O boards including one of the PCD7.R5xx flash modules.

It is also possible, equipping a PCD3.F1xx module with a PCD7.R5xx flash module.

The PCD3.R5xx and the PCD3.F1xx are not hot pluggable.

The inserted flash devices will have the following names:

SL0FLASH when a PCD3.R5xx or a PCD3.F1xx equipped with a flash device PCD7.R5xx is inserted in I/O slot #0 of a PCD3.Mxxxx.

2.5 PCD3.R600 devices with PCD7.R-SDxxx

The PCD3.R600 devices equipped with a PCD7.R-SDxxx SD card can be inserted in one of the 4 I/O slots. It is an I/O interface board including a SD card (128 MByte up to 512 MByte). From a user point of view (user program, FTP and HTTP-direct), this device will behave identically as the PCD3.R5xx/F1xx devices (but the access will be slower). The same device names are visible in e.g. the FTP client connected to the Saia PCD®.

On a SD card used on the PCD3.R600 the SBC File system SFS (using 90% of the available space on the SD card) is built on top of FAT16. If the SD card has been formatted (by a PC with a card reader) with another file system, the PCD3.R600 won't be able to read its content or to re-create a SFS.

Files stored on the SFS of a SD card (PCD7.R-SDxxx) used on a PCD3.R600 can be read and written by using the software SBC SD File system Explorer. This tool is available from the support site and is also delivered directly on the PCD7.R-SDxxx (SD card for PCD3.R600), in the PC-readable section (10% of the whole space of the card).

Please refer to the PCD3 hardware manual 26/789 for detailed operations.

3 SBC FTP Server

3.1 Introduction

FTP (File Transfer Protocol) is a commonly used protocol for exchanging files over any network that supports the TCP/IP protocol (such as an intranet).

There are two devices involved in an FTP transfer: a server and a client.

- The FTP server (e.g. a PCD3), running the FTP server task, listens on the TCP/IP network for connection requests from other devices.
- The client computer, running FTP client software (e.g. Filezilla or the Internet Explorer), initiates a connection to the server (PCD3). Once connected, the client can do a number of file manipulation operations such as uploading files to the server, download files from the server and delete files on the server (given the PCD3 is equipped with a flash device supporting file system).

Virtually every computer platform supports the FTP protocol. This allows any computer connected to a TCP/IP based network to manipulate files on a PCD3 or a PCD2.M5xxx that features a TCP/IP interface and a flash device with a file system.

There are many existing FTP client programs that can be used to connect to the PCD3 built-in FTP server. In this manual the examples are done with the following software:

- FileZilla (version 2.2.16, freeware)
- Total Commander (version 7.01)
- Microsoft® Windows Explorer
- Microsoft® Internet Explorer 6 and 7 [1])

As soon as a PCD3 or the PCD2.M5xxx has a configured IP address, its built-in FTP server can be accessed by an FTP client given the physical TCP/IP connection exists and the IP settings (network number and subnet mask) match each other.

The default user for accessing the SBC FTP Server is:

User name:	root
Password:	rootpasswd

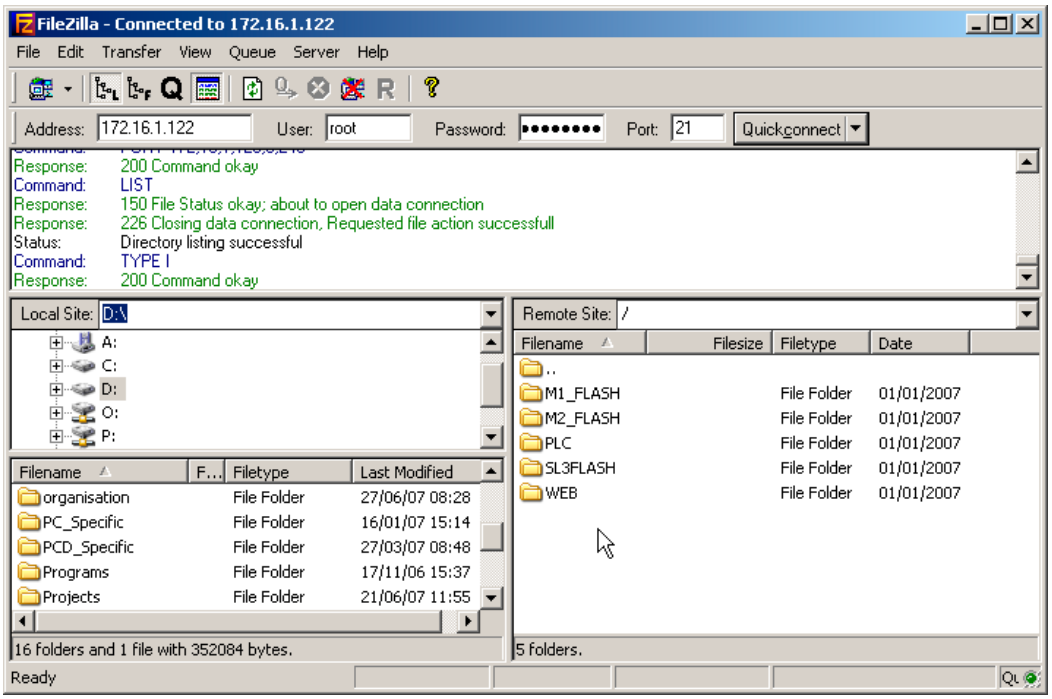
Note that only the “active mode” for FTP connections is supported on a Saia PCD®. Therefore it might be required configuring the FTP client (e.g. FileZilla) accordingly.

1) The FTP client implementation of the Windows Internet Explorer 7 (IE7) is not very well suitable for FTP operations. With the IE7 (e.g. version 7.0.5730.11) it is only possible reading files from an FTP server but it is not possible uploading files to an FTP server.

3

3.2 Drive names

Once the FTP session to the SBC FTP-Server is established, the following drives will be visible on the Saia PCD® (Mx_FLASH and SLx_FLASH are only displayed if an according flash memory module supporting file system is plugged in):



Mx_FLASH: M1_FLASH or M"_FLASH. This are the drives representing the flash card(s) PCD7.R5xx plugged in the M1 or M2 slot of the PCD3.M5xxx or of the PCD2.M5xxx. The slot number is indicated at the beginning of the drive name.

The drives contain two sub folders that are created when the drives are formatted:

- WEBPAGES — This is the folder where files for the Saia PCD® Web Server shall be placed.
- CONFIG — This folder can contain optional configuration files of the Saia PCD® Web Server and the SBC FTP-Server which are interpreted at start-up of the Saia PCD®.

SLxFLASH: e.g. SL3FLASH. This is the drive representing flash cards plugged in either a PCD3.R5xx, PCD3.F1xx or PCD3.R600. The number behind the "SL" indicates the I/O slot in which the module is plugged on the PCD3.

Also these drives contain at least the two folders WEBPAGES and CONFIG as the Mx_FLASH drives.

PLC: This drive is used for special functions of the Saia PCD® firmware (e.g. the alarming functionality). Do NOT read/write from/to this drive!

WEB: In this drive a copy of the default web pages of the Saia PCD® Web Server and the files downloaded in a DBX (created by the PG5 Web Builder) are placed. Do NOT read/write from/to this drive!

3.3 SBC FTP Server configuration

At start-up (power on) of the Saia PCD®, the FTP-Server looks in specific places for a configuration file with the name FTPConfig.txt. If such a file is found, it is interpreted and the FTP-Server is configured accordingly. This allows e.g. disabling the FTP-Server or configuring several users and their passwords.

The syntax of the configuration file is as following:

- All lines starting with a # are ignored and treated as comment
- Empty lines are skipped
- Parameter names are case sensitive
- User names, passwords and parameter arguments (e.g. on/off) are not case sensitive
- The parameter name shall appear at the beginning of a line (except spaces or tabs)
- The syntax of the parameter lines is:
  <Parameter name>=<Arguments> # Optional comment

3.3.1 Configuration parameters

The following parameters of the SBC FTP-Server configured using the configuration file.

Parameter name	Arguments	Description
FTPStart	<on/off>	**FTP Start Status** Defines whether the FTP-Server shall be started or not Default is “on”
FTPPort	<decimal value>	**FTP Default Port** Defines default FTP port where an FTP client can issue connection requests. The used value shall be carefully selected in order not to interfere with already defined ports, e.g. port 80 is used for Web Server. Default FTP port is 21
FTPMaxInstNbr	<decimal value>	**FTP Instance number** Defines how many server instances can run in parallel. Negative values, 0 and values bigger than 5 will not be taken into account. Default is 3.
FTPConnection-Timeout	<decimal value>	**FTP Connection timeout** Defines if an open connection is automatically closed after a specified number of seconds if no commands have been received. Value smaller than 30 shall be avoided, as well as values bigger than 1 day (86400). Default is that connection remains open until the FTP-Client closes its connection with the server.

Parameter name	Arguments	Description
FTPRemoveDefaultUser	<decimal value>	**FTP Default User** Defines if the default user “root” (which is “hard coded” in the FTP server) shall be kept within the internal FTP-Server user table. 1 means that it will be removed. Any other value means it is kept. By default, the entry is kept.
UserName	See next chapter	**FTP User** A number of users can be specified in the configuration file, together with some access parameters. Please refer to the next chapter for further details. By default, only the default user is configured.

3.3.2 User configuration

In order to implement a protection and access organisation of the files and folders on the file system of the Saia PCD® the following features are implemented:

- On the file system of the Saia PCD® each file “belongs” to one of eight specific groups. [1])
- If a file is created, the group that “owns” that file is defined by the GroupID of the user that creates the file.
- An FTP user can have access to files and folders of several groups. The access of these groups is specified by the parameter “AccessGroup”.
- Accessing a file is only possible if the access is done by a FTP user (or e.g. an Saia PG5® FBox) that has access rights (→ the AccessGroup involves matches the GroupID) for the group which “owns” the relevant file is placed and for the “owner” of the folder(s) in which the file is placed.
- Files within a directory can only be created if the FTP user or the Saia PG5® FBox has an AccessGroup that involves matches the GroupID.
- One FTP user can only have one group as GroupID. [2])

The definition of the FTP users (name, password, GroupID, AccessGroup, AccessType) is done in the FTP configuration file.

The definition of the GroupID and AccessGroup for file system operations executed by the user program (using System Function Calls CSFs) is defined on each call of an according System Function. When working with Saia PG5® FBoxes the according GroupID and the AccessGroup are defined in each Saia PG5® FBox.

The AccessGroup of other tasks such as the Saia PCD® Web Server are hard coded (the Saia PCD® Web Server does only have access rights for the WEB group).

The user “root” does have a special configuration as its GroupID as well as its AccessGroup permission are “All Groups” (ALLG). This means that the user “root” can access files of all groups and a file created by “root” can be accessed by all the AccessGroups (and therefore by every user).

1) Unless the file has been created by the default user „root“. In this special case the file belongs to every group (AllGroups).
2) The exception is the GroupID “AllGroups”. In this special case the file belongs to everyone (the file can be accessed by every user having any “AccessGroup” configured).

For security reasons it is recommended defining users with specific GroupIDs and GroupAccess fitting the application rather than just leaving all access rights to the one of root.

The syntax of the user definition

The syntax of the user definition in the FTP configuration file (FTPConfig.txt) is the following:

UserName=<username>,<password>,<GroupID>,<AccessGroup>[,<AccessType>]

Explanation of the parameters

The parameters of the user definition are .

<username>	is a string defined from the “=” character up to the “,” character, including space and any special characters.
<password>	is a string defined from the “,” character up to the next “,” character, including space and any special characters.
<GroupID>	is a value specifying the user group, e.g. a file created by this user will belong to that <GroupID>. The <GroupID> is to be given as hexadecimal value.
<AccessGroup>	is a value specifying the file system access group(s). A file / directory will be accessible if its <GroupID> belongs to the given <AccessGroup>. The <AccessGroup> is to be given as hexadecimal value.
<AccessType>	is the kind of access given to the user, either RD_ONLY (it is not possible to delete / nor write into a file nor create a file in any provided file system) or RD_WR. <AccessType> is an optional parameter and is set to RD_WR by default.

Existing GroupIDs

On the Saia PCD® file system eight specific groups are defined. Three of them are reserved for specific purposes (CONFIG, DOWNLOAD and WEB).

The other groups (USER1 to USER4) can be used by the Saia PCD® programmer for its own purposes.

- The WEB group is reserved for the Web Server task of the Saia PCD®. It is only possible writing files into the WEBPAGES folder when the AccessGroup involves the GroupID WEB. The Web Server task does read all files from with any GroupID (AccessGroup=AllGroup).
- In order to access the CONFIG folder on the flash devices, the user needs to have rights for the CONFIG group (for configuring the HTTP direct access and the FTP server). The Saia PCD® does read configuration files with any GroupID (AccessGroup = AllGroup).
- The DOWNLOAD group is reserved for future firmware features and should not be used.
- The groups USER1 to USER4 can be used by the Saia PCD® programmer for its own purposes.

The valid GroupID values are defined as following:

0x02 User belongs to CONFIG Group

0x04 User belongs to DOWNLOAD Group

0x08 User belongs to WEB Group (can be accessed by the Saia PCD® Web Server task)

0x10 User belongs to USER1 Group (free group)

0x20 User belongs to USER2 Group (free group)

0x40 User belongs to USER3 Group (free group)

0x80 User belongs to USER4 Group (free group)

0x00 This is special a GroupID indicating that the files created by this user can be accessed by everyone. This value has been kept for compatibility with “root”, but it should be avoided when specified in a configuration file.

Only values specified in the table above can be used as GroupID values!

AccessGroups

The definition of the AccessGroup value is calculated by a logical OR operation on all the GroupIDs that shall be accessible by the FTP user or the System Function call. If for example a FTP user shall have access to the files owned by group USER1, USER3 and group WEB, the following operation is done:

Name of the group	hexadecimal representation		binary representation
WEB	0x08		0000 1000 binary
USER1	0x10	OR	0001 0000 binary
USER3	0x40	OR	0100 0000 binary
Resulting AccessGroup:	**0x58**		0101 1000 binary

In fact each group is indicated by a bit (according to the value of the GroupID) in one byte (representing the AccessGroup value):

USER4 0x80	USER3 0x40	USER2 0x20	USER1 0x10	WEB 0x08	DOWN-LOAD 0x04	CONFIG 0x02	reserved bit
No access = 0	Access = 1	No access = 0	Access = 1	Access = 1	No access = 0	No access = 0	Always = 0

AccessType

By providing the optional parameter AccessType (possible values: RD_ONLY or RD_WR), the access of a user can be limited to “Read Only”. Note that this parameter will automatically apply to all the groups this user can access. It is not possible providing write access to one group and read only access to another group for one single user.

The AccessType is an optional parameter and is set to RD_WR if it is not provided in the user definition.

3.3.3 Configuration file location

The configuration file for the FTP-Server task is to be placed in one of the following folders. If such a file exists in more than one location, only the first found will be considered (the folders are checked in the same order as this list).

- INTFLASH:/Config (integral on-board Flash)
- WEB:/WEBPAGES (User program)
- M1_FLASH:/Config (Flash device in M1 slot)
- M2_FLASH:/Config (Flash device in M2 slot)
- SL0FLASH:/Config (Flash device in I/O slot #0)
- SL1FLASH:/Config (Flash device in I/O slot #1)
- SL2FLASH:/Config (Flash device in I/O slot #2)
- SL3FLASH:/Config (Flash device in I/O slot #3)

For the flash file system (e.g. M2_FLASH:/Config), the file is downloaded using the FTP-Server (and the currently used parameters).

Note that WEB drive (WEB:/WEBPAGES/) it is the representation of a DBx as part of the Saia PCD® user program. In order to place your FTPConfig.txt to the folder WEB:/WEBPAGES/, just copy it into the html folder of your PG5 Project and add it to the Web Builder (*.wsp) file of your PG5 project.

Downloaded parameters will not be taken into account right after the download of the file but only at next power on of the Saia PCD®.

The priority of the phases, where the configuration file is searched, has been changed with firmware version 1.08.23 on PCD3 systems. Older firmwares checked in the following sequence:
M2_FLASH:/Config,
WEB:/WEBPAGES,
M1_FLASH:/Config,
SLxFLASH:/Config (x for 0…3)

3.3.4 Configuration file example

The following lines provide an example of configuration file “FTPConfig.txt” which can be downloaded to the PLC.

```
# ****************************************
# FTP Configuration file
#
# Default values
#       FTPStart=on
#       FTPPort=21
#       FTPMaxInstNbr=3
#       FTPConnectionTimeout=0          0=No timeout, != 0 timeout of specified seconds
#       FTPRemoveDefaultUser=0          Default user and password is kept
#       UserName=root,rootpasswd,0x00,0xFE,rd_wr
#
# ****************************************
#
# Uncomment next line do forbid FTP connections
# FTPStart=off
#
# ****************************************
#
# Overwritten values
FTPPort=6034                            # Check if this value is NOT used by any other con-
nections
FTPMaxInstNbr=2                         # Two instances max
FTPConnectionTimeout=3600               # 1 hour timeout if no command received
FTPRemoveDefaultUser=1                  # default user is removed
UserName=newuser,hello,0x10,0xFE        # User = newuser
                                        # password = hello
                                        # Belong to Group USER1.
                                        # Have access to all files / directories
                                        # By default read/write access
UserName=root,12hrs37,0x00,0xFE         # User = root
                                        # password = 12hrs37
                                        # Does not belong to a specific group
                                        # Have access to all files / directories
                                        # By default read/write access
UserName=limited,,0x80,0xC0,rd_only     # User = limited
                                        # no password is defined
                                        # Belong to Group USER4
                                        # Have access to files / directories belonging to
                                        #               USER3 and USER4 (1100’0000) groups
                                        # defined with read only access
```

3.4 Examples for connecting the SBC FTP Server

This chapter contains examples for establishing the SBC FPT Server using various standard FTP clients. Also some software specific hints are provided.

Because there is no information about the creation date or the last modification date of a file on the SBC File system, the values shown by the FTP clients are not relevant (always the same default value).

Also the GroupID is not represented correctly as the Saia® File System does not work with the standard users and groups.

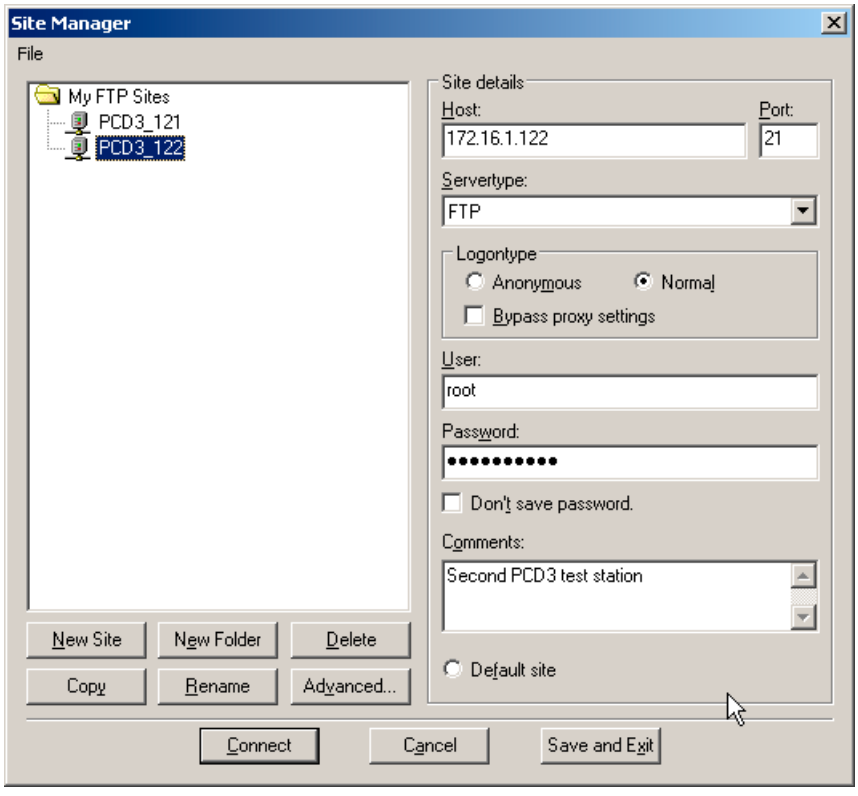
3.4.1 FileZilla

FileZilla is a popular and free FTP client software for Windows® (and other OS platforms) written by Tim Kosse that can be downloaded the following homepage http://sourceforge.net/projects/filezilla.

The only point to be considered for connecting the SBC FTP-Server using FileZilla is that the “passive mode” is set by default and needs to be changed to “active mode”.

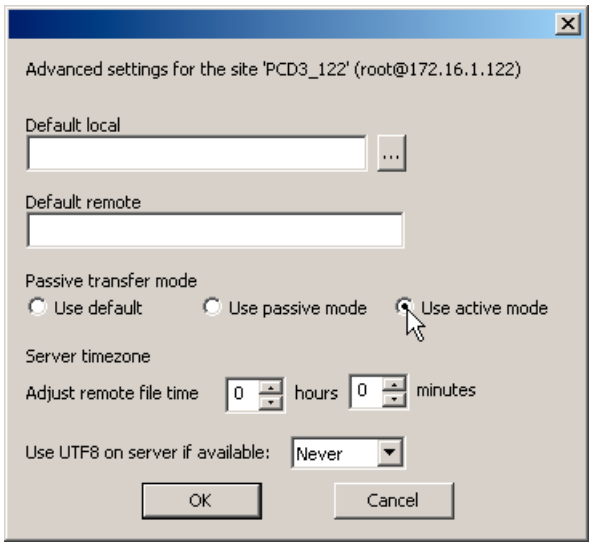
In order to configure FileZilla for connecting the SBC FTP Server, execute the following steps.

- Open FileZilla
- In the menu “File”, select “Site Manager…”
- In the window “Site Manager”, click the button “New Site”
- Enter a name for your connection and fill in the host address, port, server type and so on as shown in the screenshot below:

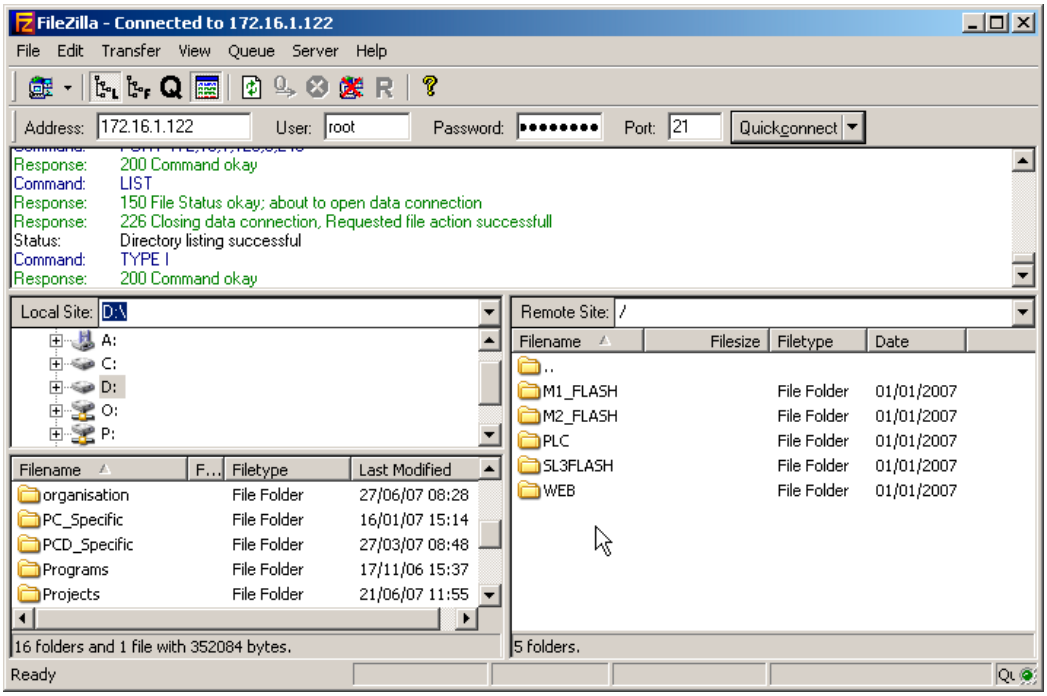


The password for the default user “root” is “rootpasswd”.

- Click the button “Advanced…”
- Select the radio button “Use active mode” and click “Ok” for closing the window.



- Back in the “Site Manager”, click the button “Connect” (or double click the station) for connecting the SBC FTP Server:



- Now it is possible browsing the folders on the PCD3 FTP Server.
  - A file can be downloaded from the PCD3 FTP-Server by double clicking it on the right side of the FileZilla Window (it will be downloaded to the folder indicated on the left side; Local Site).
  - A file or folder from the PC can be uploaded by right-clicking it on the left side of the window and selecting “Upload”.
- For disconnecting, select “Disconnect” from the menu “File”

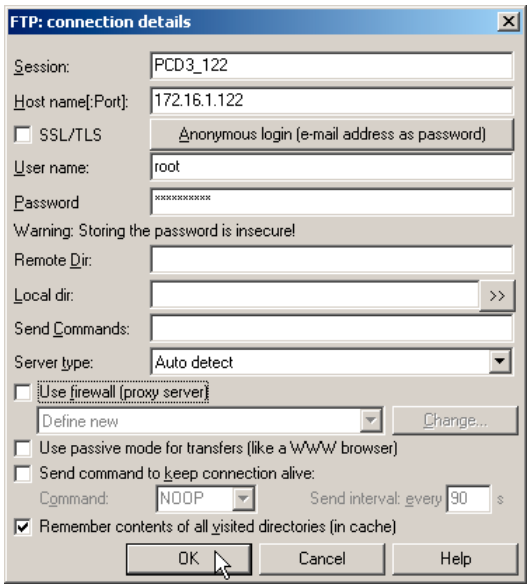
It is recommended always properly disconnecting the session. If this is not done, the connection will remain open on the server. In this case it will only be possible connecting three times (afterwards no new connections will be accepted as the maximum number of parallel connection is 3 by default).

3.4.2 Total Commander

The Total Commander is a Shareware file manager for Windows® 95/98/ME/NT/2000/ XP/Vista and Windows® 3.1. It can very well be used for comfortably transferring files from and to a SBC FTP Server.

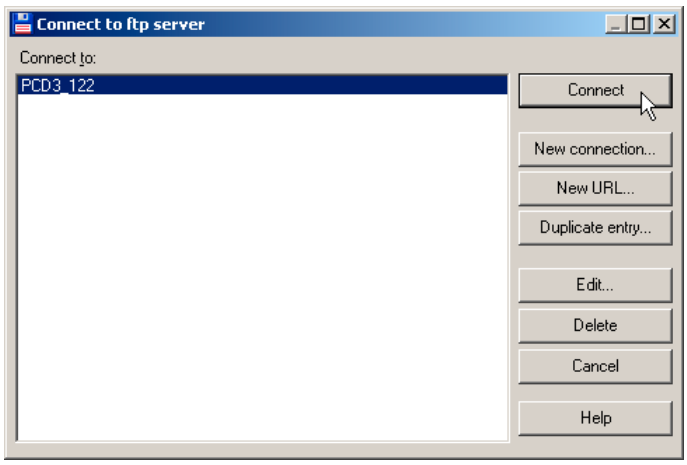
In order to configure an FTP connection to a PCD3, follow the following steps.

- Open the Total Commander
- From the menu “Net”, select “FTP Connect…” and in the appearing window, click the button “New Connection…”.
- Enter the name for the session, the IP address of the SBC FTP-Server as well as the user name with the password and click ok.



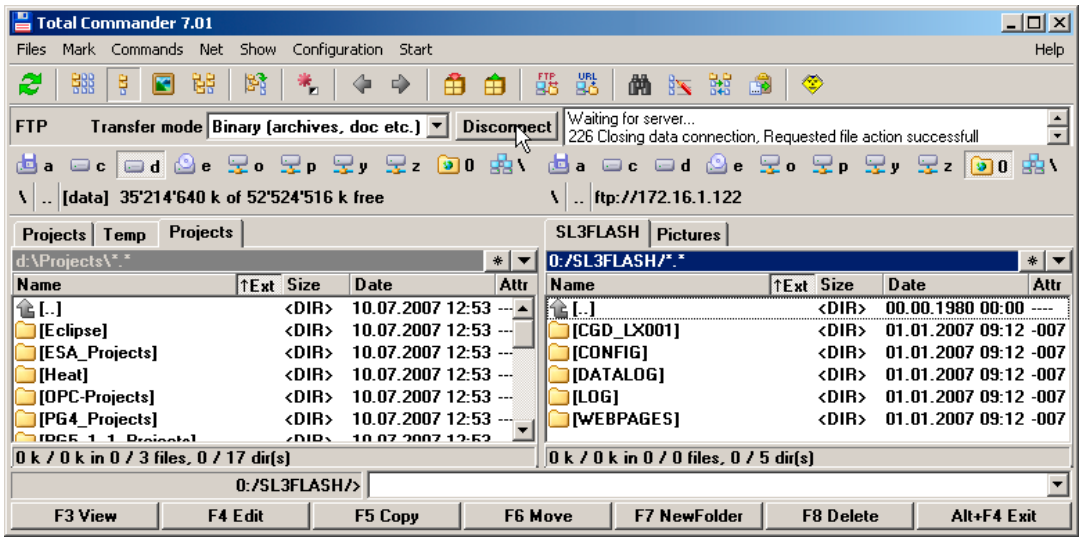
The default password for the user “root” is “rootpasswd”.

- Back in the window “Connect to ftp server”, select the FTP server to be connected and click the button “Connect”.



- Once connected, the content of the FTP server is shown on one side of the Total Commander. The files can be copied from and to the server by selecting the file and then clicking the button “F5 Copy” or just hitting the key F5.

The status of the operation are indicated in the window appearing just aside of the “Disconnect” button of the FTP connection.



- For properly disconnecting the FTP connection, click the button “Disconnect”.

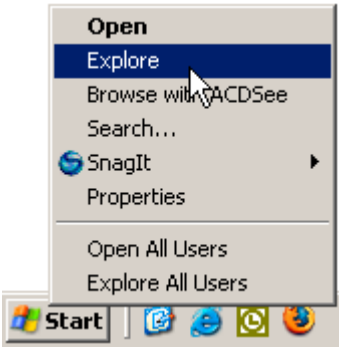
Note that double clicking e.g. a *.csv file on the server will download this file to the cache and then open it from the cache. If saved, the file will only be saved in the temporary directory on your PC but not on the Saia PCD® file system. In order to update a file on the PCD3, save it locally, edit it and then copy it back onto the server.

3.4.3 Windows® Explorer

The Windows® Explorer can be as FTP Client as well. The advantage of the Windows® Explorer is that this explorer does support drag-and-dropping files and folders. The handling of the Windows® Explorer is similar to the handling of the Microsoft® Internet Explorer 6.

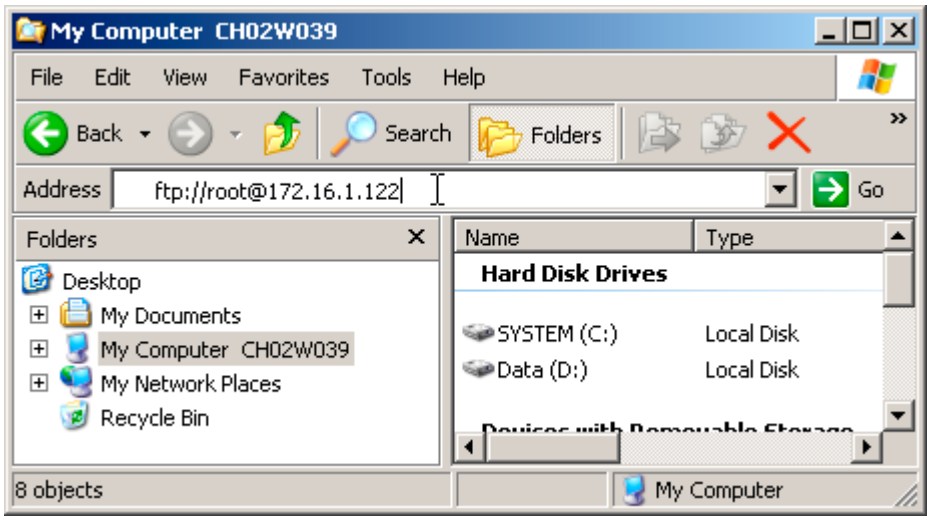
In order to use the Windows® Explorer for accessing the SBC FTP Server, execute the following steps:

- Open the Windows® Explorer (right-click the Windows Start button and select “Explore”)



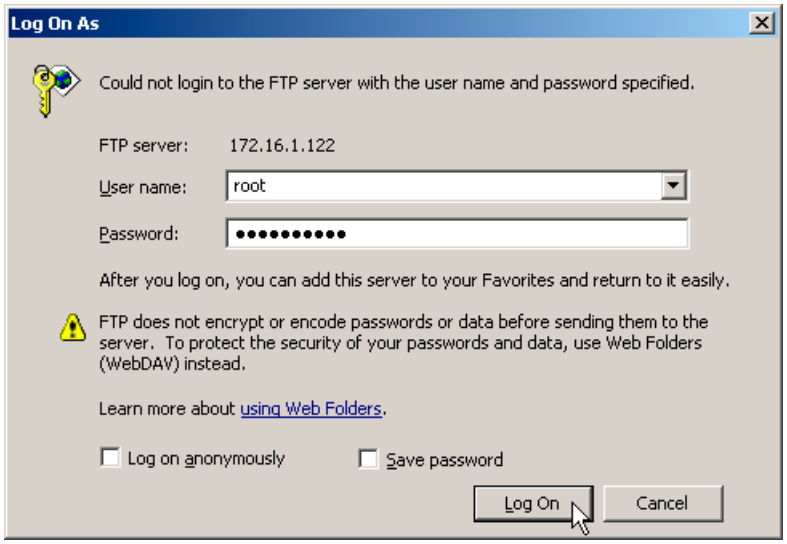
- In the address field type the following address (don’t forget the “ftp:// at the beginning):

  ftp://<username>@<IP address> [1])



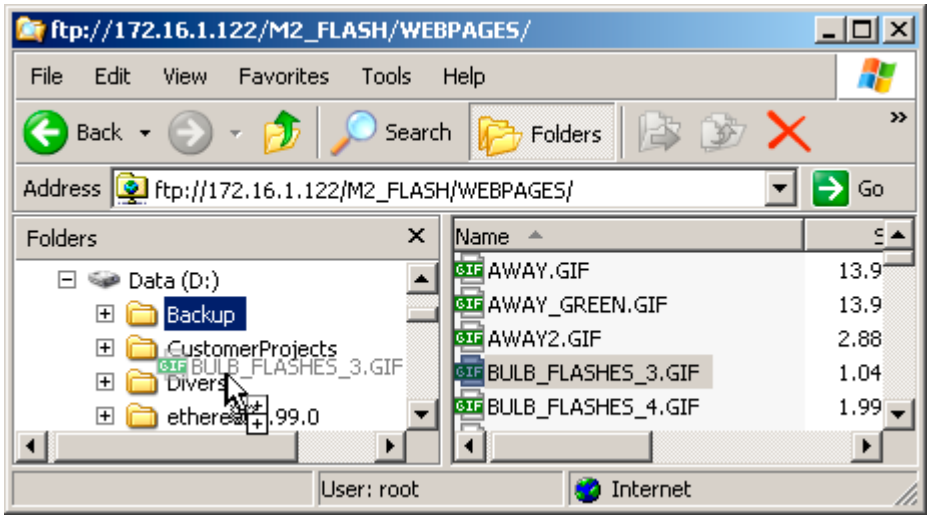
- In the appearing “Log On As” window, provide the user name and the password and click “Log On”

1) The “root@” is used to provide the user name that will log in. It is also possible not providing the user name in the address. If doing so, a message will pop up that you are not logged in. After closing this message, right-click on the right area of the IE and select “Login as…” in order to get to the login window.



The password for the default user “root” is “rootpasswd”.

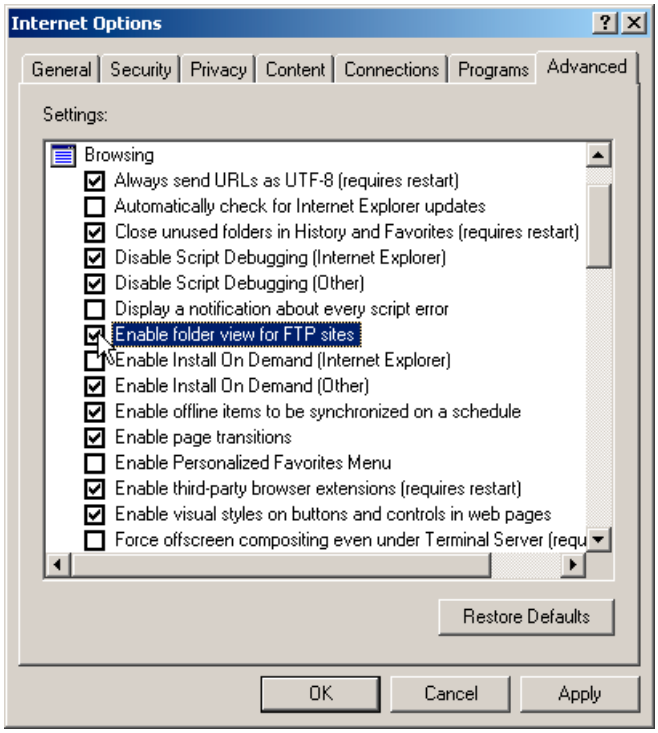
- Once logged in, you can browse and edit folders on the SBC FTP Server. Files and folders are downloaded and uploaded by drag-and-dropping them.



Note that double clicking e.g. a *.csv file will download this file to the cache and then open it from the cache. If saved, the file will only be saved in the temporary directory on your PC but not on the Saia PCD® file system. In order to update a file on the PCD3, save it locally and then drag-and-drop it back to the folder from where you opened it.

For enabling the drag-and-drop functionality in the IE6 (or the Windows® Explorer) for FTP sites the according option needs to be configured (“Enable folder view for FTP sites”). If this is not done, it will not be possible to view and manage the files and folders on the SBC FTP Server.

The according option can be found in the menu “Tools”, “Internet Options”



It is not possible properly disconnecting an FTP session with the Internet Explorer because the IE does not feature a disconnect functionality. As result, the connection remains open also if the IE is closed or the PC is shut down. This leads to the fact that by default it is only possible to connect the Saia PCD® three times (→ until the maximum amount of open FTP connections is reached). Afterwards the Saia PCD® needs to be powered off and on again in order to accept new connections.

If you plan using only the IE or the Windows Explorer for connecting the PCD3 FTP Server, it is strongly recommended you are configuring a timeout on the FTP-Server of the PCD3 (see chapter “FTP-Server configuration).

3.4.4 Internet Explorer 6 and 7

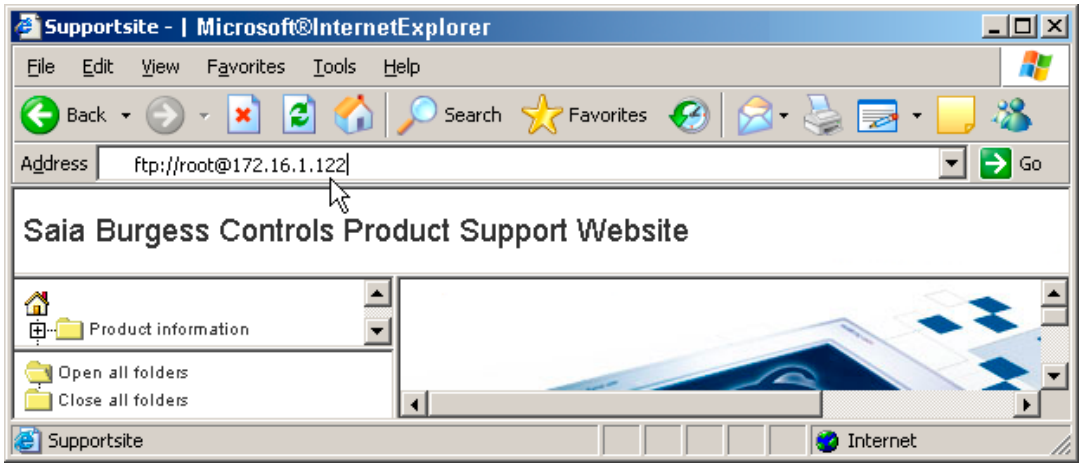
It is possible using the Microsoft® Internet Explorer 6 or 7 for connecting the SBC FTP Server. For doing so, executed the following steps:

Internet Explorer 6

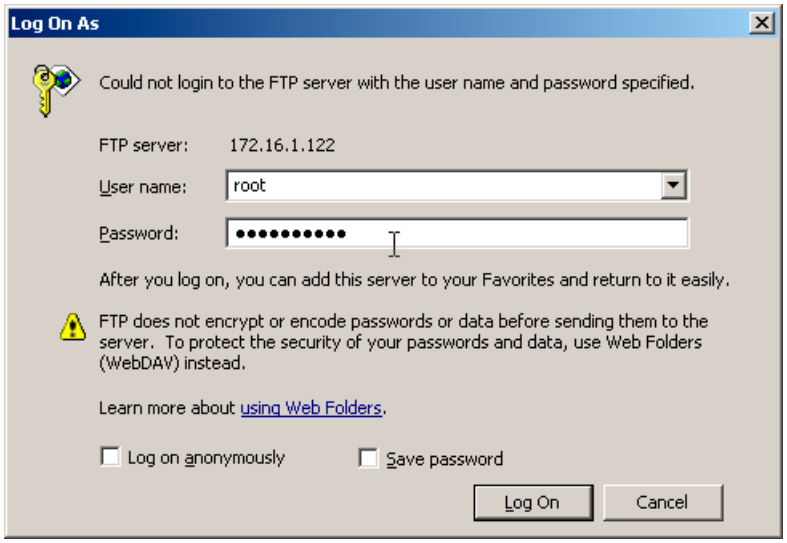
- Open the IE6 and type the following address (don't forget the "ftp:// at the beginning):

  ftp://<username>@<IP address> [1])

After entering the address, click <Enter>



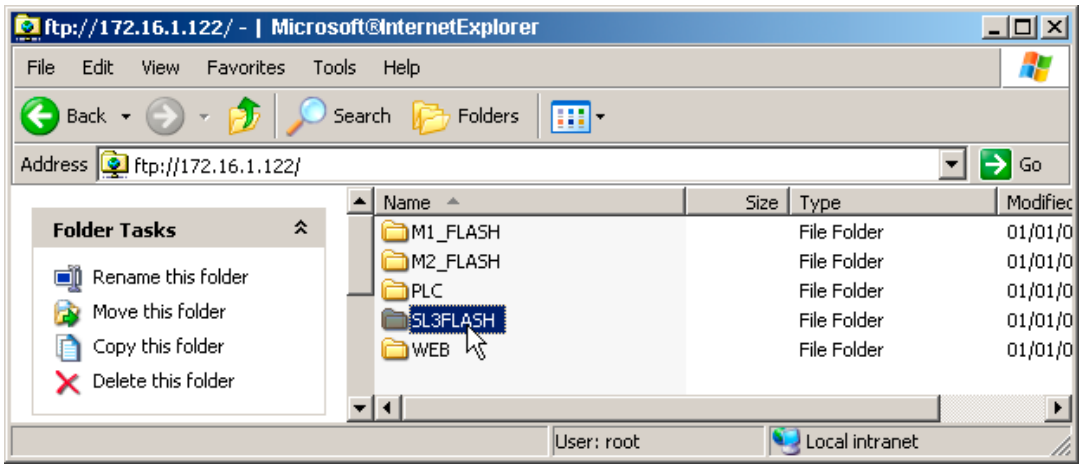
- In the appearing "Log On As" window, provide the user name and the password and click "Log On"



The password for the default user "root" is "rootpasswd".

1) The "root@" is used to provide the user name that will log in. It is also possible not providing the user name in the address. If doing so, a message will pop up that you are not logged in. After closing this message, right-click on the right area of the IE and select "Login as…" in order to get to the login window.

- Once logged in, you can browse and edit folders on the SBC FTP Server. Files and folders are downloaded and uploaded by drag-and-dropping them.



Note that double clicking e.g. a *.csv file will download this file to the cache and then open it from the cache. If saved, the file will only be saved in the temporary directory on your PC but not on the Saia PCD® file system. In order to update a file on the PCD3, save it locally and then drag-and-drop it back to the folder from where you opened it.

For enabling the drag-and-drop functionality in the IE6 (or the Windows® Explorer) for FTP sites the according option needs to be configured (“Enable folder view for FTP sites”). If this is not done, it will not be possible to view and manage the files and folders on the SBC FTP Server.

The according parameter is set in the menu “Tools”, “Internet Options” (see the example with the Windows® Explorer)

It is not possible properly disconnecting an FTP session with the Internet Explorer because the IE does not feature a disconnect functionality. As result, the connection remains open also if the IE is closed or the PC is shut down. This leads to the fact that by default it is only possible to connect the PCD3 times (→ until the maximum amount of open FTP connections is reached). Afterwards the Saia PCD® needs to be powered off and on again in order to accept new connections.

If you plan using only the IE for connecting the PCD3 FTP Server, it is strongly recommended you are configuring a timeout on the FTP-Server of the PCD3 (see chapter “FTP-Server configuration).

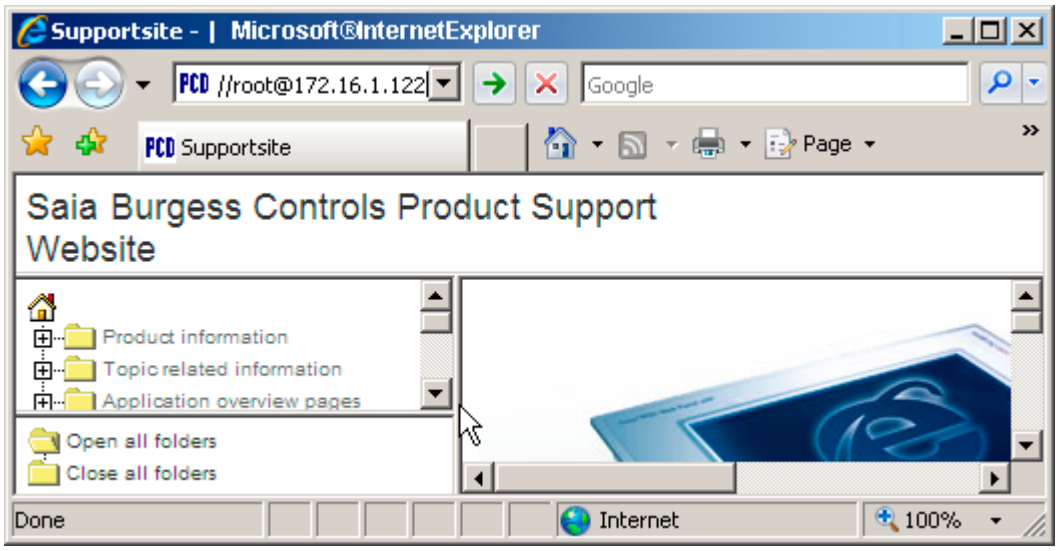
Internet Explorer 7

Also the Microsoft® Internet Explorer 7 can be used for connections to the Saia PCD® built-in FTP server. Unfortunately it seems not being possible uploading files to the PCD3 as it is no longer possible drag-and-dropping files into the window of the IE7 (and by doing so uploading the files to the server).

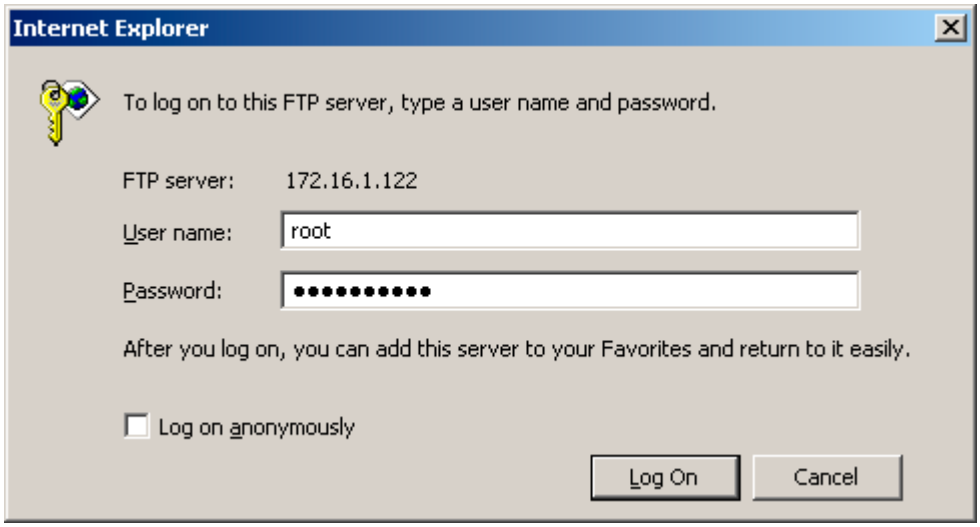
A second problem is that the Internet Explorer 7 does not “remember” the user that logs on the server (e.g. “root”), this information needs to be specified each time a folder is changed or a file is downloaded. Follow the steps below in order to view the content and download files from the PCD3:

- Open the IE6 and type the following address (don't forget the "ftp:// at the beginning):
  ftp://<username>@<IP address>

After entering the address, click <Enter>



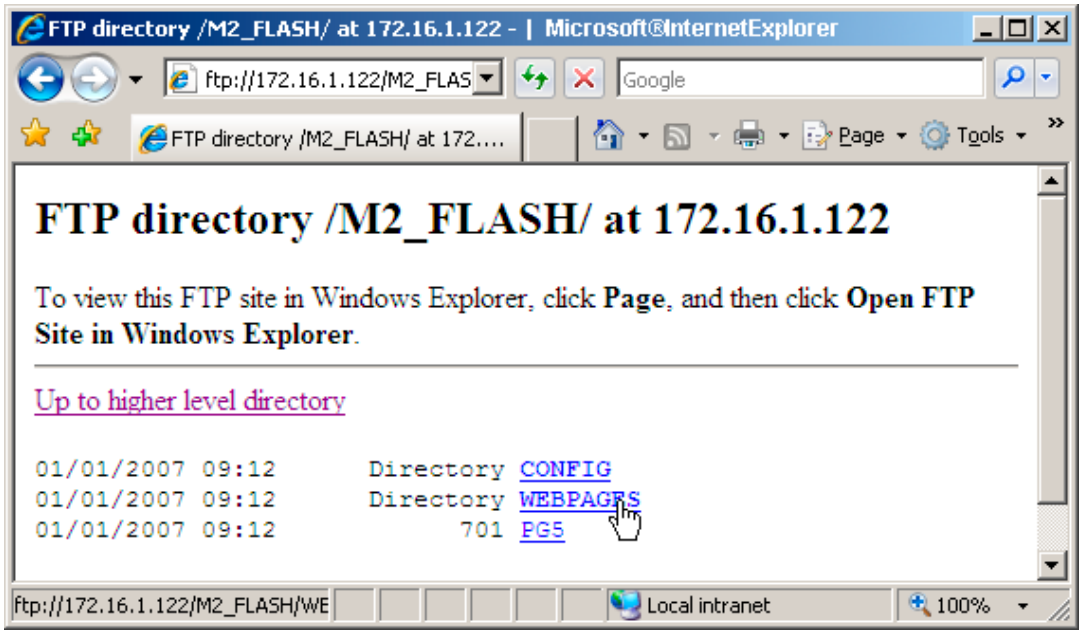
- In the appearing "Log On As" window, provide the user name and the password and click "Log On"



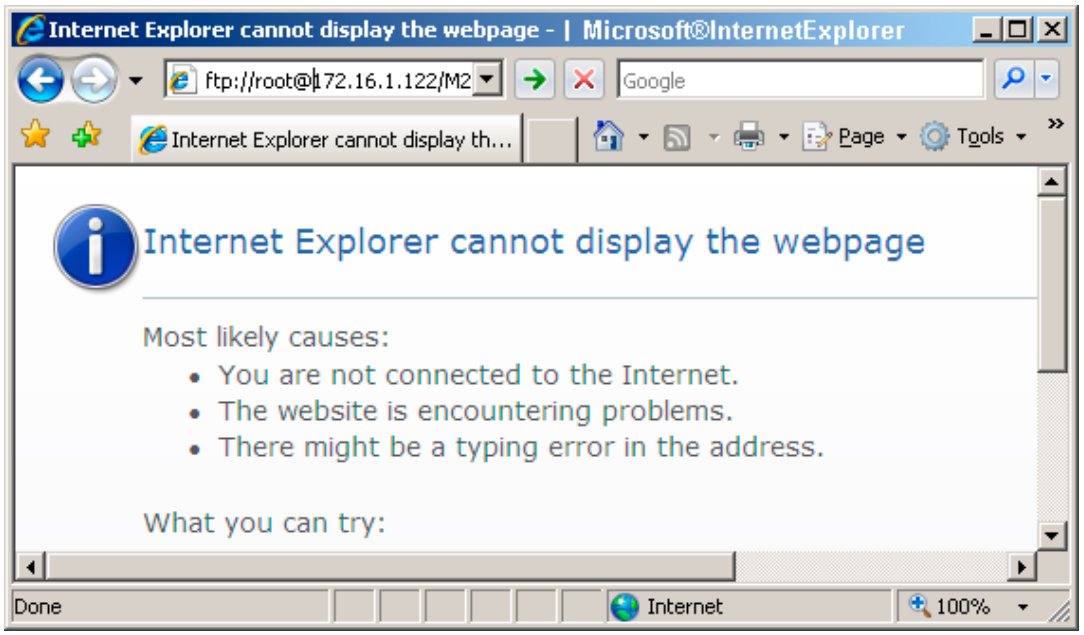
The password for the default user "root" is "rootpasswd".

- Once logged in, you should see a similar view than the one below. Instead of the folder icons as in IE6, the IE7 does only provide a list of the folders and files present on the server.

1) The "root@" is used to provide the user name that will log in. It is also possible not providing the user name in the address. If doing so, a message will pop up that you are not logged in.



- You can change a folder or download a file by clicking on the according link. When doing so, the following massage will appear:



In this view you have to insert the “root@” in the address bar for changing to the according directory. After this is done and the Enter key is pressed, the according folder will be shown or you will be prompted for storing the according file.

It is not possible properly disconnecting an FTP session with the Internet Explorer 7 because the IE does not feature a disconnect functionality. This leads to the same restriction as described in the description of the Windows® Explorer.

3.5 Note regarding firewalls

FTP is an unusual service in that it utilizes two ports, a “data” port and a “command” port (also known as the control port). Traditionally these are port 21 for the command port and port 20 for the data port. The confusion begins however, when we find that depending on the mode, the data port is not always on port 20.

In active mode FTP the client connects from a random unprivileged port (N > 1023) to the FTP server‘s command port, port 21. Then, the client starts listening to port N+1 and sends the FTP command PORT N+1 to the FTP server. The server will then connect back to the client‘s specified data port from its local data port, which is port 20.

This constellation is not foreseen in commonly used firewalls implemented in e.g. ADSL routers (on the client side). The configuration of such a specific firewall requires specialized knowledge and can not be covered in this manual. Please contact your network administrator for further information and the configuration of your firewall.

4 File system management using CGI calls

4.1 Introduction

It is possible to visualize the state (device state, used blocks etc.) of the flash file system(s) of a Saia PCD® using the PCD Web Server. It is also possible to see the current flash state (busy compressing, busy formatting, device error etc). Further on it is possible to start a file system compression or a (re-)format procedure.

These features can be accessed by using the CGI (Common Gateway Interface) of the Saia PCD® Web Server together with specific tags described in the following table.

4.2 Module addressing

The different flash modules and their file systems are addressed by the following values. In the table below the numbers from this list are to be placed instead of the “x”.

- 0: related to M1_Flash: device
- 1: related to M2_Flash: device
- 2: related to SL0Flash: device
- 3: related to SL1Flash: device
- 4: related to SL2Flash: device
- 5: related to SL3Flash: device

Tag name	R/W	Description
NT-FileSys,DeviceStatus[x]	Read only	Provides the following device status: “DeviceName:” Device NOT mounted “DeviceName:” Device mounted with error(s) “DeviceName:” Device mounted OK
NT-FileSys,DeviceActionStatus[x]	Read only	Provides the following device status: “DeviceName:” Action not started “DeviceName:” Compressing... “DeviceName:” Formatting... “DeviceName:” Action finished OK “DeviceName:” Action finished with error(s) “DeviceName:” Action NOT executed, e.g. parameter error “DeviceName:” Action Undefined “DeviceName:” Device is currently busy
NT-FileSys,DeviceBlockSize[x]	Read only	Provides the following device status: “DeviceName:” <current device block size>
NT-FileSys,DeviceBlockNbr[x]	Read only	Provides the following device status: “DeviceName:” <current number of blocks on the device>
NT-FileSys,DeviceSize[x]	Read only	Provides the following device status: “DeviceName:” <device size>

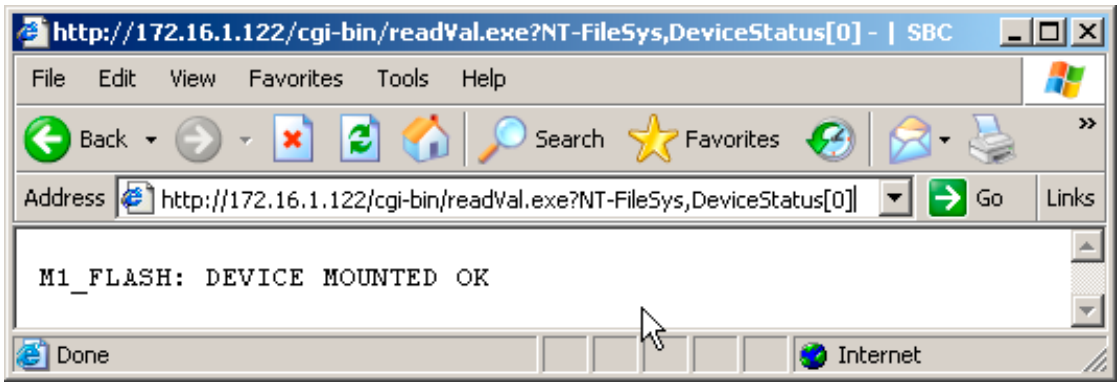
Tag name	R/W	Description
NT-FileSys,DeviceReqBlockSize[x]	Read/Write	Provides the following device status: “DeviceName:” <new/current block size> New block size can be given as argument 512 .. 8 KB for PCD7.R5xx devices (in steps of *2) 4 KB .. 512 KB for PCD3.R600 devices (in steps of *2)
NT-FileSys,DeviceReqBlockNbr[x]	Read only	Provides the following device status: “DeviceName:” “new/current block number” The new values is calculated according to new provided block size.
NT-FileSys, StartDeviceCompression[x]	Read/Write	Provides the following device status: “DeviceName:” 0: no compression active “DeviceName:” 1: compression active When setting this value to ‘1’, a compression is started on the selected device. The DeviceAction-Status will return the status of the compression.
NT-FileSys,StartDeviceFormatting[x]	Read/Write	Provides the following device status: “DeviceName:” 0: no formatting active “DeviceName:” 1: formatting active “DeviceName:” 2: formatting active When setting this value to ‘1’, a reformatting is started if the current device status is unknown or mounted with error, or if a new block size has been provided in the DeviceReqBlockSize parameter. When setting this value to ‘2’, a reformatting is started in any case. The values of the Device-ReqBlockSize is taken as block size (new or identicial to current block size). Other values are ignored. The DeviceActionStatus will return the status of the compression.
NT-FileSys, EnableAutoCompression[x]	Read/Write	Reading this tag allows to know the current status of the automatic compression. “0”: automatic compression is disabled “1”: automatic compression is enabled It is possible to configure the automatic compression, either through an user program system function call or by writing a “0” or a “1” with the tag.
NT-FileSys,DeviceUsedBlockNbr[x]	Read only	Returns the size really used on the device, in the form: “DeviceName:” <used size> Used blocks includes internally used blocks, file (data) used blocks and blocks released (e.g. when a file is deleted) but not yet recovered by a call to the compression algorithm.
NT-FileSys,DeviceFreeBlockNbr[x]	Read only	Returns the free size of the device, in the form: “DeviceName:” <free size>
NT-FileSys,DeviceFreedBlockNbr[x]	Read only	Returns the total size of freed blocks of the device, in the form: “DeviceName:” <freed size>

4.3 Example using the Internet Explorer

The fastest way to visualize the state of a flash device using the Saia PCD® Web Server is just entering the according CGI call with the corresponding tag in a web browser like the Internet Explorer. The syntax for the call is the following:

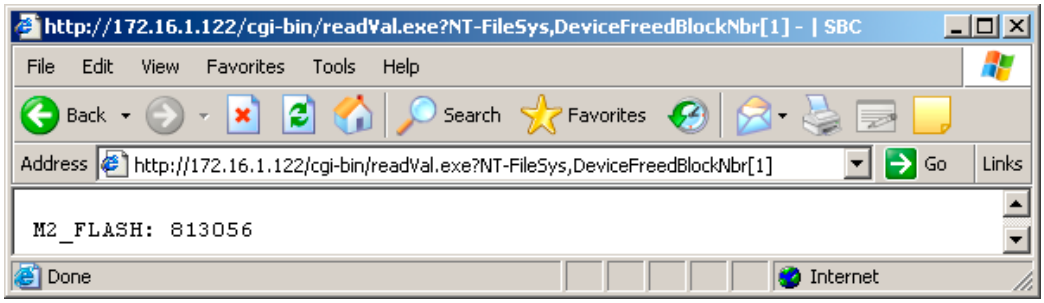
http://<IP-Address of the PCD>/cgi-bin/<CGI command>?<file system specific tag>[,<optional …>,<optional parameter>]

For reading the state of the status of the flash device mounted in the slot M1 (→ address 0) the call would look like below:

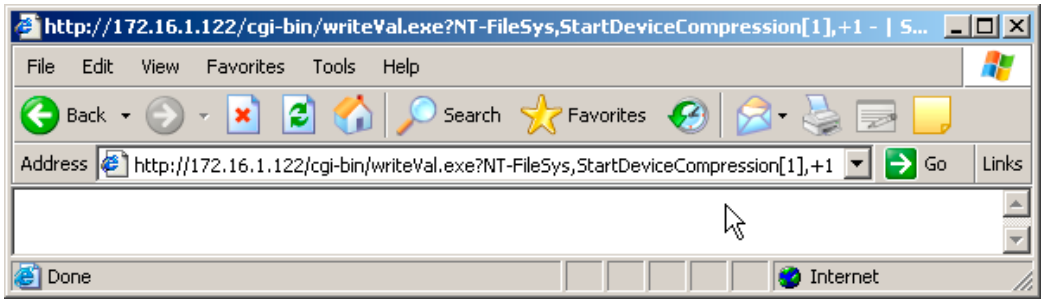


In case a writeable tag is to be written, the CGI command “writeVal” instead of the “readVal” will be used. The following example shows the sequence for checking the amount of freed blocks of the device mounted in slot M2 (→ address 1), launching a compression and then verifying the reduced amount of freed block:

- Read the amount of freed blocks on the flash device mounted in the slot M2:

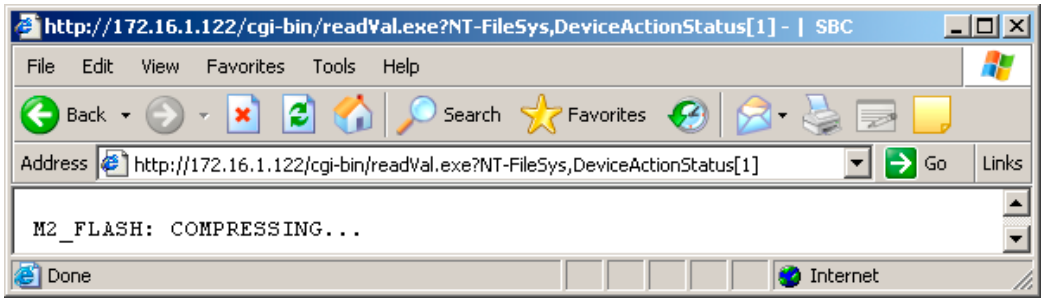


- Start a compression of the flash device mounted in slot M2:

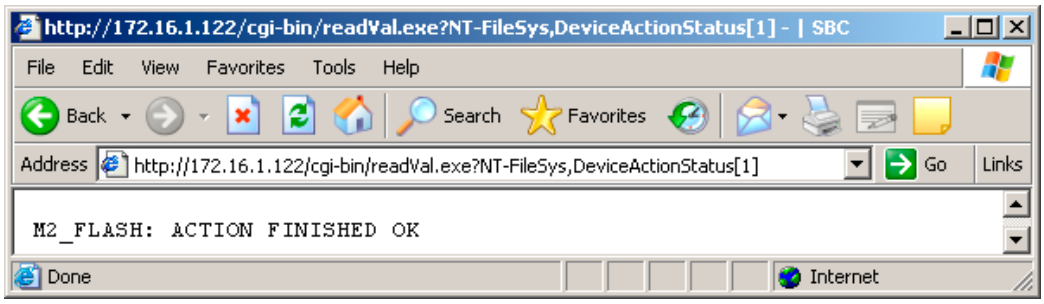


(There is no return value to the start of the compression).

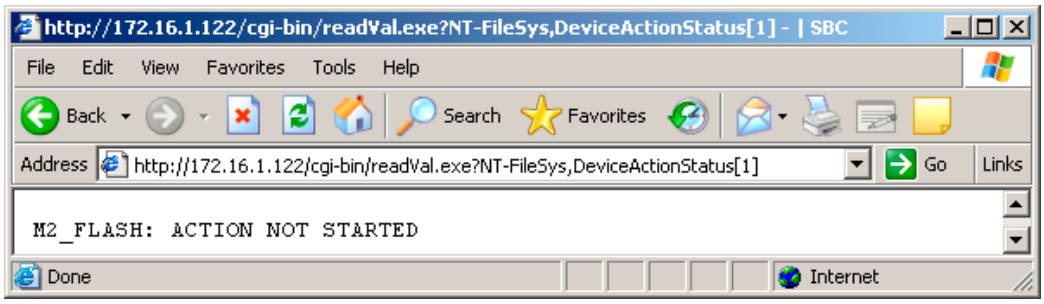
- Check the current action executed on the device mounted in slot M2:



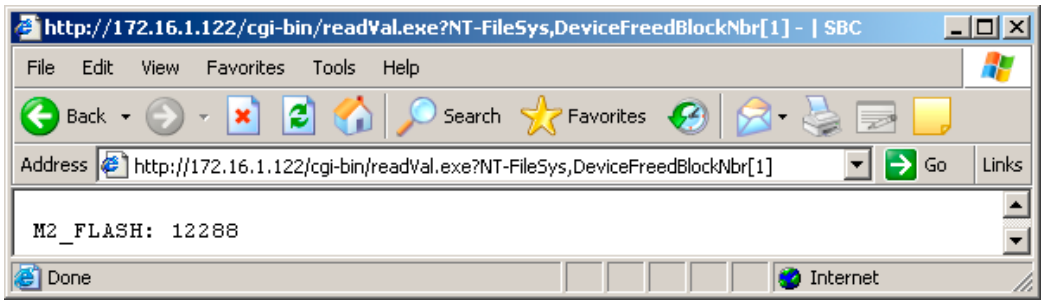
- The same call will give the feedback once the compression has been finished:



Note that this message will only be shown once. The second call of the Device-ActionStatus will indicate “ACTION NOT STARTED”



- Check again whether the amount of freed blocks has been reduced (note that it is normal that few blocks will remain freed).



The example above is just a demonstration of the file system management on a Saia PCD® using CGI calls. Of course it is far more comfortable using these calls directly from an application (e.g. Visual Basic) or from a web page.

The SBC Web Editor (version 5.11.06) does not support the usage of the file system specific tags of the Saia PCD®.

A Appendix: File system Saia PG5® FBoxes

A.1 Introduction

The file system can be accessed and managed by the user program. For a comfortable usage of the System Functions supporting the file system a dedicated FBox library is available.

In this chapter the FBoxes are presented and shortly described. Please refer to the Online Help of the FBoxes for further information.

A.2 Working principle

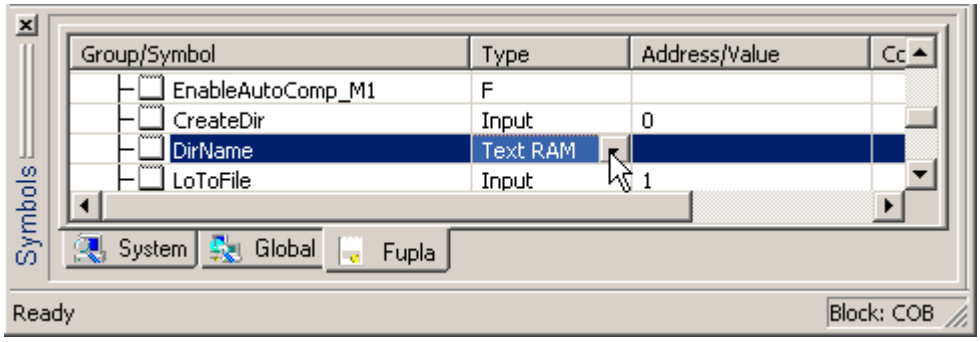
For each flash device supporting file system, one “Memory Management” FBox for the file system is to be placed before any other FBox accessing the relevant flash device. This Management FBox has to have a project wide unique name. By this name other FBoxes from the file system library are referring to the Master FBox (and thereby to the relevant flash device).

A.3 Providing directories and file names

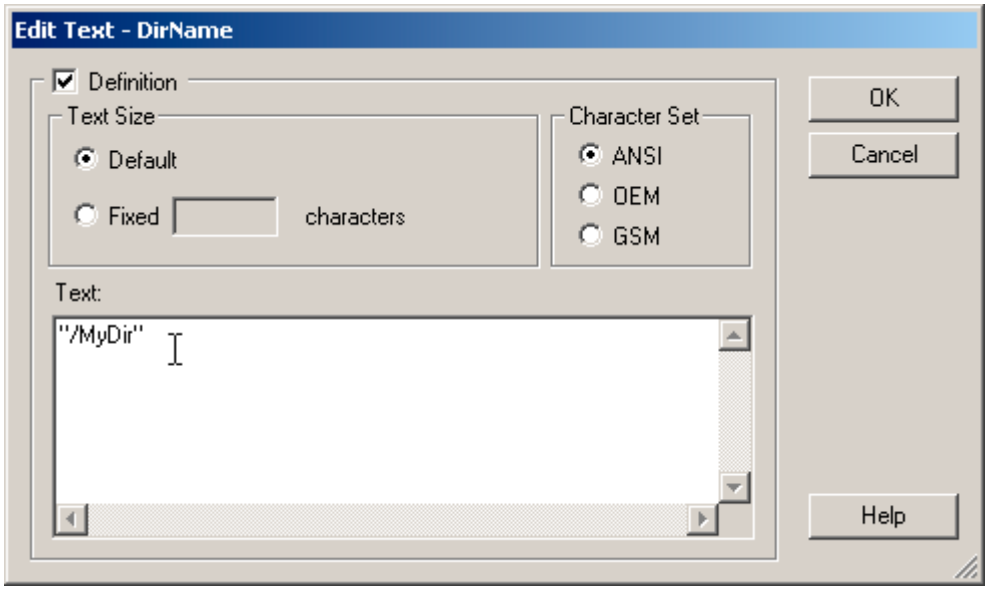
File- and path names are passed as Saia PCD® text to the FBox. The according Saia PCD® text is to be entered on the face of the FBox.

Not the symbol name but the content of the text is relevant for the FBox. The following steps are required for defining a text with e.g. the file name “MyFile” (the extension like e.g. *.txt will automatically be appended by the file name according to the parameters in the Memory Management FBox).

- In the Symbol Editor, create a new symbol and define the type as “Text” (or “RAM Text” if the content shall be modified in run time):



- Double click the small ☐-sign in front of the symbol name.
- In the appearing window, define the text content, e.g. “MyDir”.



Directory names

If e.g. a new directory is created with the FBox “Create Directory” the directory name (e.g. “/NewDir”) is to be passed in a Saia PCD® text to the FBox. If a sub directory in this directory is created, the subdirectory is to be passed as “/NewDir/SubDir” to FBox (in a new call as the FBox can not create several directories in one call).

File names

Also the file names are provided in the same way, but in this case there is no leading backslash required (e.g. “Filename”). In the “Memory Management” it is possible specifying an optional extension (e.g. *.txt) that will be appended to all created files.

A

It is also possible selecting the option “Extension=None” in the FBox “Memory Management” for disabling the automatically added extension. In this case consider that the index would be added to the extension if you specify an extension and have the the index enabled (see below).

Indexed files

If a file is created it is also possible appending an index to the file name (“MYFILE” resulting in MYFILE007.TXT). The index value is passed as input value to the FBox. The shown digits of the index are to be specified in the “Memory Management” FBox.

Summary

With the FBoxes of this library, file names are defined as follow:

- The memory name is selected in the Management FBox or in the Property FBox.
- The directory and subdirectory names are defined in a Saia PCD® text.
- The file name is defined in another Saia PCD® text.
- The index value is taken from an FBox input.
- The Index length is defined in the Management FBox or in the Property FBox.

- The file extension is defined in the Management FBox or in the Property FBox.
- The memory, the file and the extension separators are inserted automatically.
- The directory and sub-directory are optional but recommended. If used, a slash must be entered as first character in the Saia PCD® text.

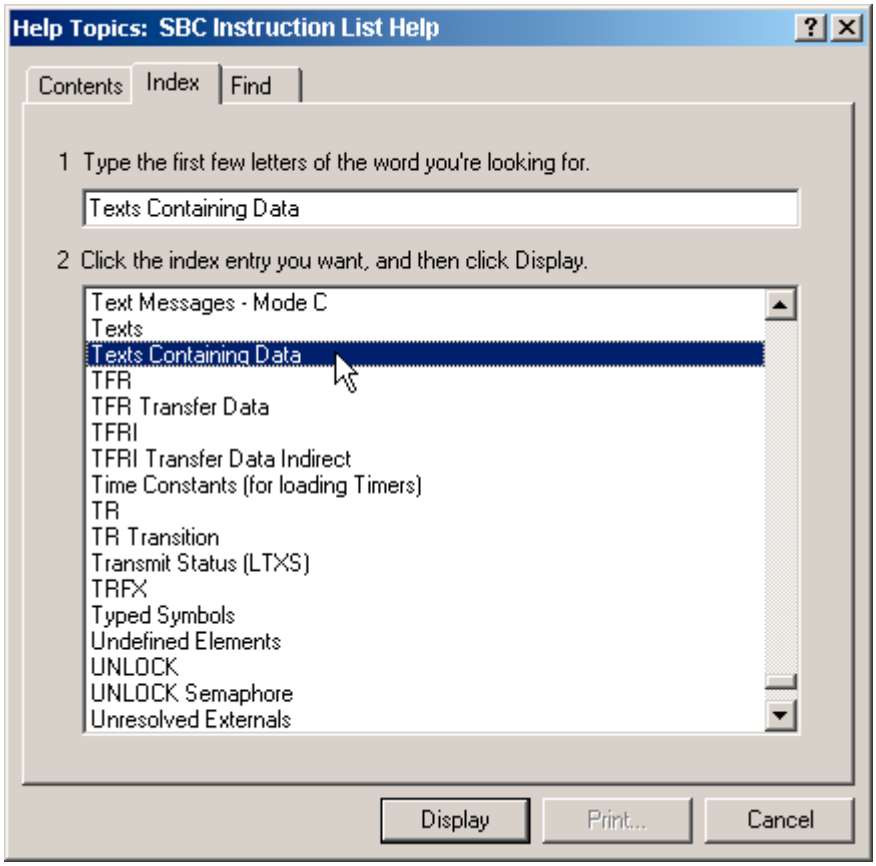
  Example: „/files“

  or „/files/data“

The following points are to be considered when working with the FBoxes for the file system:

- Maximum length of files or directory names (dot and extension included) is 23 characters.
- Maximum file length including the path (and including 10 characters for memory name) is 63 characters.
- Flash devices, directory names and file names are always converted to upper case by the Saia PCD®.

Template texts (containing $ and @) can also be used in directory and file names, e.g. for introducing the current date or a value of a register. For further information please refer to the Online Help of the SBC Instruction List Editor SEdit (open the help, click the button “Index”, type in “text” and double-click the entry “Texts containing data”).



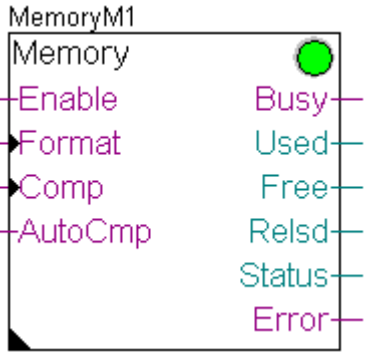
The help file can be launched from SEdit directly or from the installation folder of Saia PG5® (SEdit32.hlp).

A

A.4 The existing Saia PG5® FBoxes

Below the FBoxes for accessing and working with the file system are described. For more detailed information please refer to the Online Help of the specific FBox.

A.4.1 The “Memory Management” Saia PG5® FBox



The “Memory Management” FBox must be placed before any other FBox that accesses the relevant flash device. In this FBox the GroupID for all created files on the flash device as well as the AccessGroup for the operations (create, delete, rename, read, write etc.) are configured.

The “Memory Management” FBox does provide information regarding the corresponding flash device such as:

- Used space on the flash device in kBytes (including the freed memory that can not be re-written until the device has been compressed)
- Available memory in kBytes
- Released (freed) memory in kBytes

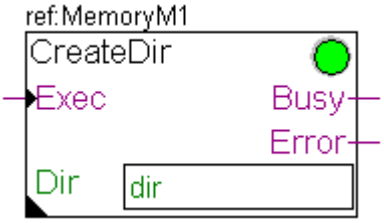
Further on this FBox allows compressing or formatting the relevant flash device. It is also possible to enable or disable the automatic compression (without the use of this FBox, the firmware of the Saia PCD® does decide when a compression task is started).

A.4.2 The “File Properties” Saia PG5® FBox

The FBox “File Properties” can be used to modify the configured values of the GroupID and the AccessGroup of the FBoxes accessing a flash device with file system.

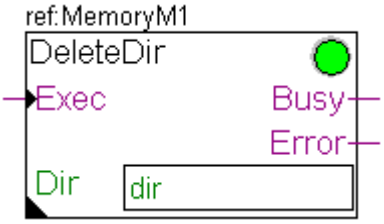
Also the file name format (extension and/or index) etc. that are configured in the FBox “Memory Management” can be overwritten with this FBox.

A.4.3 The “Create Directory” Saia PG5® FBox



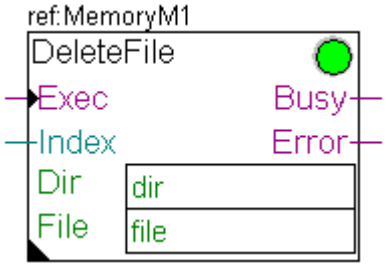
This FBox creates a directory on the file system. The full path of the directory must be entered in a Saia PCD® text (e.g. “/NEWDIR” or “/NEWDIR/SUBDIR”).

A.4.4 The “Delete Directoy” Saia PG5® FBox



This FBox deletes a directory on the on the file system, including any file in it. The name of the directory must be entered in a Saia PCD® texts (e.g. “/NEWDIR” or “/NEWDIR/SUBDIR”).

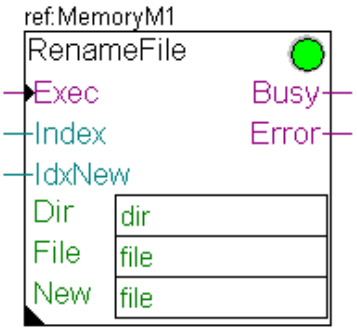
A.4.5 The “Delete File” Saia PG5® FBox



This FBox deletes a file on the on the file system. The full path of the directory and the file name must be entered in 2 Saia PCD® texts (e.g. “/DIRECTORY” and “FILE-NAME”). The file name can be extended by an index.

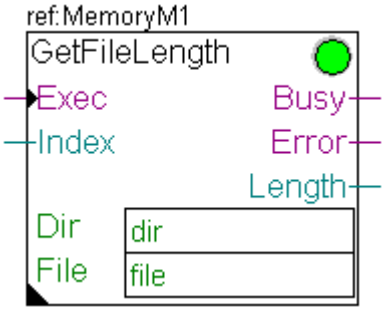
A

A.4.5 The “Rename File” Saia PG5® FBox



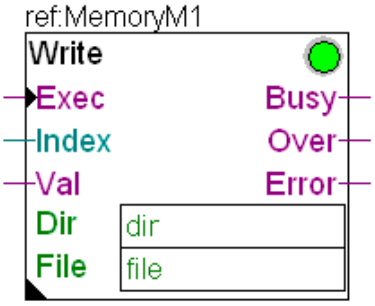
This FBox renames a file on the on the file system. The full path of the directory, the file to rename and the new file names must be entered in 3 Saia PCD® texts. The original and the new file names can be extended by an index. The original file gets the common file properties of the referenced FBox. The properties of the new file name (index and extension) can be adjusted in the FBox.

A.4.6 The “Get File Length” Saia PG5® FBox



This FBox allows the user to get the actual length of a file on the file system. The full path of the directory and the file name must be entered in 2 Saia PCD® texts. The file name can be extended by an index.

A.4.7 Saia PG5® FBoxes for writing data to a file



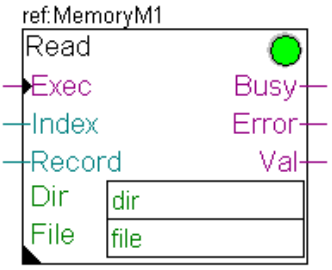
Write a record from a Saia PCD® media (Flag, Input, Register etc.) to a file. There are three FBoxes with that functionality (“Write Binary”, “Write Integer” and “Write Float”).

This FBoxes write data at the end of the specified file on the file system. The full path of the directory and the file name must be entered in two Saia PCD® texts. The file name can be extended by an index. The file is automatically opened before and closed after the write operation.

If a write command is given while the memory is busy, the data are temporarily stored

in an internal buffer (in the FBox and not on the flash memory). The buffer is automatically copied to the memory as soon as it is available again. The user needs to estimate and adjust the necessary buffer length for his application. If a new command is detected when the buffer is full, the command is ignored and the overflow output shows the error.

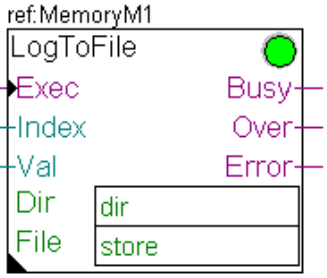
A.4.8 Saia PG5® FBoxes for reading data from a file



Read a record from a file to a Saia PCD® media (Flag, Output, Register etc.)

This FBox reads data from the specified file at the specified record position. The full path of the directory and the file name must be entered in 2 Saia PCD® texts. The file name can be extended by an index. The file is automatically opened before and closed after the read operation.

A.4.9 Saia PG5® FBoxes for creating log files



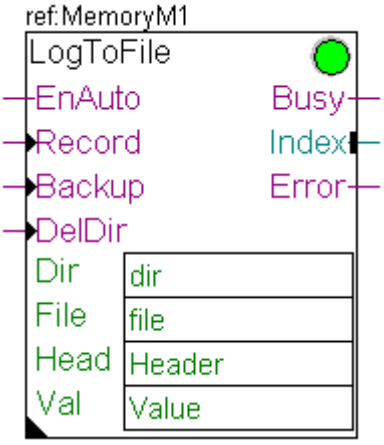
Log to file

This FBoxes writes data in plain text into the specified file on the file system. The full path of the directory and the file name must be entered in two Saia PCD® texts. The file name can be extended by an index. The file is automatically opened before and closed after the write operation.

Written values are separated by the separator defined in the common file properties. If selected, the date and/or time is inserted at the beginning of the line. A carriage return and a line feed are added at the end of each record. With this FBox, it is possible creating called *.csv files (comma delimited files). CSF files can be opened and handled by spread sheet applications like Microsoft® Excel.

If a write command is given while the memory is busy, the data are temporarily stored in an internal buffer (in the FBox and not on the flash memory). The buffer is automatically copied to the memory when it is available again. The user needs to estimate and adjust the necessary buffer length for his application. If a new command is detected when the buffer is full, the command is ignored and the overflow output shows the error.

Log to file advanced



This FBox writes data in plain text into the specified file on the file system. The recording can be automatic at predefined interval.

The full path of the directory and the file name must be entered in 2 Saia PCD® texts. The file is automatically opened before and closed after the write operation. At predefined interval an automatic backup of the file is created and a new file is started.

The main use of this FBox is the following:

Data are recorded at regular interval (e.g. every ten minutes) in a temporary text file. At regular interval (e.g. once a day) the file is backed up and a new file is started. Each new backup file gets a new name created with an index or the current time. Backup files are copied and removed over FTP. The temporary file should not be accessed by FTP.

A

It is important to regularly remove the backup files in order to avoid that the file system is filled up with data.

If date and/or time is used, it is important to ensure that the clock is adjusted and working correctly. When the time is adjusted (backward), take care that backup files do not already exist. The date and the time is added with underscores between date and time but no other separator. Take care that the total file name length will not exceed 23 characters, extension included.

Example

The file ‚DATA.TXT‘ the 31 December 2007 at 12:34:56 is backed up as “DATA071231_123456.TXT” (21 characters!)

To start a clean recording, it is recommended to give a first pulse on input “Delete Directory”. This will create an empty directory, initialize the recording file and restart the index for backup files. Therefore, it is recommended to reserve a directory for one file and it’s backup.

A header text can be defined that will be written on top of each file. It allows putting a description with start date and time of the created file. Additionally an identification of each field (column) of the CSV file can be introduced. Another Saia PCD® text is used to define the data to record in the file (by using a template text). The user can build the text using the special text command ($ and @). The necessary characters at the end of each line (Carriage Return and Line Feed) are also to be edited in this tem-

plate text (<CR> adds a Carriage Return and <LF> adds a Line Feed).

If the index is selected for backup file name, its length is automatically adapted to support the maximum index value. Leading zeros are added.

Example

Highest possible index = 9	Index is 1 digit
Highest possible index = 128	Index is 3 digit

B Appendix: System Functions (Instruction List) to access file system from user program

It is possible to use system functions to access the file system(s) from the user program.

B.1 Introduction

B.1.1 Purpose of this chapter

The File System SFB Library allows accessing the different file systems created on a Saia PCD® by specific function blocks.

The library includes the following file functionalities:

- creation (file or directory)
- opening a file
- closing a file
- deletion (file or directory)
- reading an open file
- writing an open file
- seeking into an open file
- Opening / reading / closing in one call
- Opening (creating of not existing) / writing at end of file / closing in one call
- Renaming a file
- Formatting a flash device
- Compressing a flash device
- Getting device information, device status, device sizes (used, free, released, total)
- Enable / Disable automatic compression.
- Get file entries within a given directory

Some of the functions are synchronous (started, called and finished within the initial call) or asynchronous (same function shall be called many times until a return code tells it is finished).

B.1.2 The File System

The File System is an internal data storage which can be accessed through this API.

The data is located on a flash device and remains passive until the user program or another task (FTP-Server or Web-Server) needs to update, add or work with file information.

The essential file system usage steps are:

1. create, open a file by means of its name
2. optionally seek a position in the file
3. write data from the Saia PCD® context (e.g. text, DB, Register, Flags, Clock

etc.), to the file

4. read data to the Saia PCD® context (e.g. text, DB, Register, Flags, Clock etc.), from the file
5. close the file

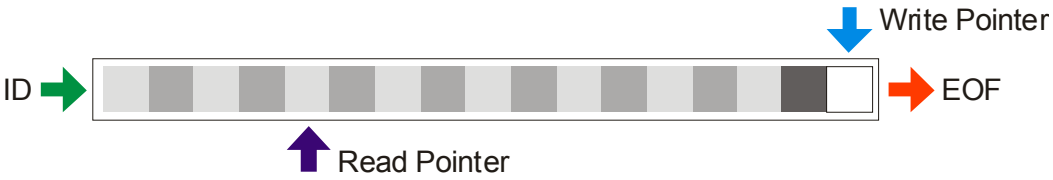
Additionally it is possible to delete a file, a directory (and all files contained in it) or inquire its size.

By opening or creating a file you receive an identifier. The identifier is hooked with two pointers:

- a read position pointer
- a write position pointer

The pointers are modified by calling read, write or the seek function.

When 4 bytes of data are read and one byte is written at the end of the file, the pointers will have following position:



The seek function modifies the read/write pointers relative to current pointer position.

A seek position of 0 resets the corresponding pointer to the beginning of the file.

Note, that the read and the write pointer can be modified. However, modifying the write pointer doesn‘t make sense as it is not possible modifying data in a file. It is only possible appending data at the end of file.

B.2 File System SFC Library

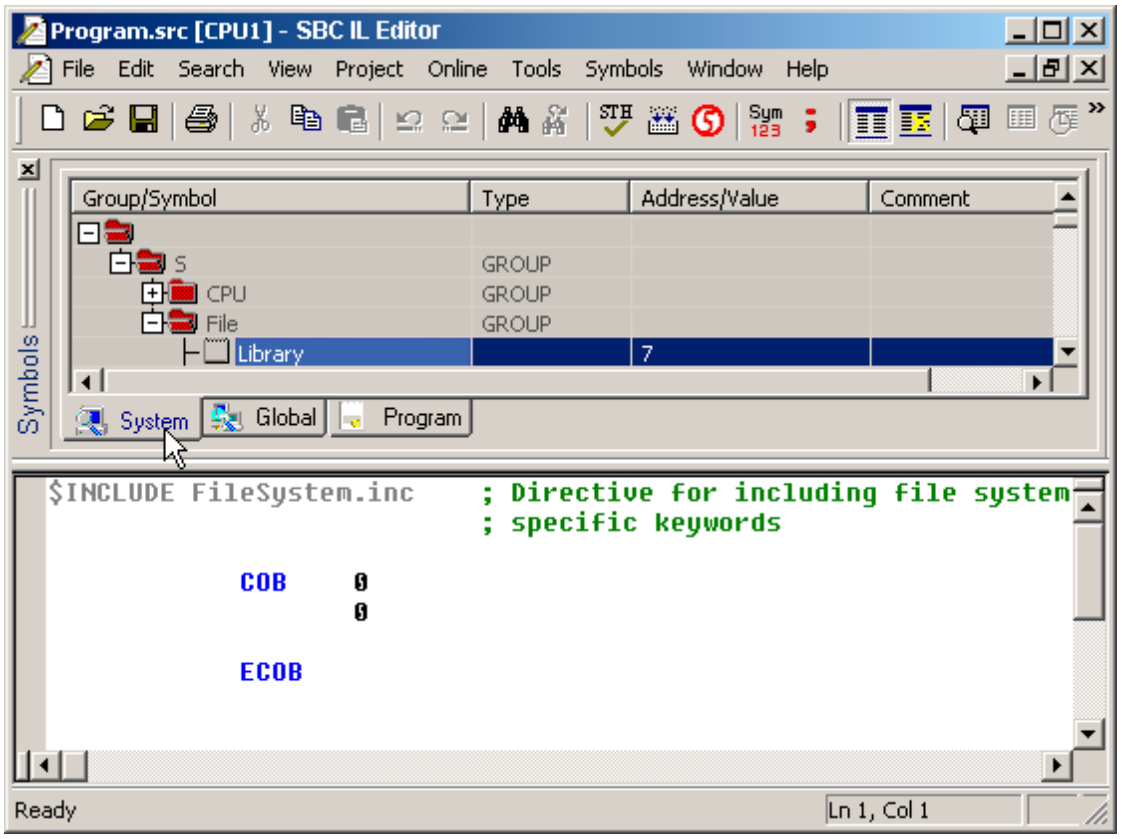
B.2.1 Introduction

The system functions for accessing a file system in a user program are accessed by the CSF (Call System Function) command. The general form of the CSF call is:

```
CSF [cc]        lib_number      ;library number 0-4095/4096-8191
                func_number     ;function number to call
                [parms...]      ;optional parameters
```

The “lib_number” and “functions_numbers” are required to execute CSF commands. Keywords can be used by including the “FileSystem.inc” file in Saia PG5®. Both numbers can be entered as keywords, by using the symbols provided by the file “FileSystem.inc”.

Note that the keywords will not be available right after writing the line, but after the next build of project. After the build, the keywords will appear in the “System-tab” of the Symbol-Editor. From there, they can be drag and dropped into the program file.



File System library calls are accessed through the library keyword:

```
S.File.Library
```

The functions available are explained in the following chapters.

B

B.2.2 Parameters

The following parameters are used by the System-Functions for the file “FileSystem.inc”.

Name	Used	Description
FileName	S.File.Create S.File.ASCreate S.File.CreateDir S.File.ASCreateDir S.File.Open S.File.Delete S.File.ASDelete S.File.SeqWrite S.File.ASSeqWrite S.File.SeqRead S.File.ASSeqRead S.File.StoreString S.File.ASStoreString S.File.FileRename S.File.ASFileRename S.File.GetIndexedFileProp	The file name must be passed as absolute path-name, e.g. “M1_FLASH:/Report.txt”. The file name may contain only alphanumeric characters (without SPACE) and “:”. Special characters like umlauts are not recom-mended. A filename or directory name can not exceed 23 characters, including extension and the total length of a passed absolute filename shall not exceed 63 characters.
GroupId	S.File.Create S.File.ASCreate S.File.CreateDir S.File.ASCreateDir S.File.SeqWrite S.File.ASSeqWrite S.File.StoreString S.File.ASStoreString	Defines to which group the created file / direc-tory belongs. One file / directory belongs to one and only one group identifier. S.File.GroupId.CONFIG S.File.GroupId.DOWNLOAD S.File.GroupId.WEB S.File.GroupId.USER1 S.File.GroupId.USER2 S.File.GroupId.USER3 S.File.GroupId.USER4 S.File.GroupID.ALLG [1]) Remark on previous versions: [1]) The symbol S.File.GroupID.ALLG is missing in the “FileSystem.inc” up to PG5 V1.4.200. If this symbol is required, use the value 255 instead of the symbol.
GroupAc-cess	S.File.Create S.File.ASCreate S.File.CreateDir S.File.ASCreateDir S.File.StoreString S.File.SeqWrite S.File.ASSeqWrite S.File.StoreString S.File.ASStoreString S.File.Delete S.File.ASDelete S.File.FileRename S.File.ASFileRename S.File.GetIndexedFileProp	This parameter allows to access files / directo-ries belonging to one of the given groups. Any combination of the above defined group can be defined. Creating a file or a directory is only pos-sible within a directory with a group belonging to the combination of given groups. Deleting a file / directory is only possible if files / directories / sub-directories and files belonging to subdirecto-ries with a group belonging to the combination of given groups. Possible group access is as defined above in the GroupId parameter. In addition, the **S.File. GroupAccess.ALLG** is defined to gain access to all groups.

B

Name	Used	Description
AccessType	S.File.Open S.File.Seek	Defines the access type when opening a file or when seeking into a file Valid values are: S.File.AccessType.RD_ONLY S.File.AccessType.WR_ONLY S.File.AccessType.RD_WR If a file is open with the Read only attribute, any calls to write data in the file will fail. If a file is open with the write only attribute, any calls to read the file will fail. When seeking into a file, the attribute determines which pointer is updated (refer to 1.2 for pointer description)
SeekPos	S.File.Seek	Seek in current open file relative to the current position (positive or negative) expressed as doubleword. A value of 0 reset the pointer(s) to the beginning of the file.
Length	S.File.Write S.File.SeqWrite S.File.ASSeqWrite S.File.Read S.File.SeqRead S.File.ASSeqRead	Number of elements to write or read. For TEXT, one element corresponds to 1 byte, whereas for DB or registers, one element corresponds to 4 bytes. Writing/reading a register with length equal to 1 will write/read 4 bytes to/from the file. The maximum value read/written in text medias is 256 bytes, in DB medias, is 64 elements (64 x 4 bytes) and with registers, is 64 elements (64 x 4 bytes).
Buffer (source)	S.File.Write S.File.SeqWrite S.File.ASSeqWrite	The source buffer may be DB, Register or TEXT elements. The maximum length transferred for a text is 256 bytes and 64 elements for DB and registers. A DB or register element contains 4 bytes.
Buffer (destination)	S.File.Read S.File.SeqRead S.File.ASSeqRead	The destination buffer may be DB, Register or TEXT elements. The maximum length transferred for a text is 256 bytes and 64 elements for DB and registers. A DB or register element contains 4 bytes.
Offset (source)	S.File.Write S.File.SeqWrite S.File.ASSeqWrite	The offset from the start of the source buffer from where to start reading data. The remaining data buffer to read from must contain length-byte, length-db elements, length register elements otherwise CSF_FS_DATA_MISS is returned.
Offset (destination)	S.File.Read S.File.SeqRead S.File.ASSeqRead	The offset from the start of the destination buffer from where to start writing data read from the file. The remaining data buffer to write to must contain length-byte, length-db elements, length register elements otherwise CSF_FS_DATA_MISS is returned.
FileOffset	S.File.SeqRead S.File.ASSeqRead	The bytes offset from where to start reading in the file.

Name	Used	Description
WrAttr	S.File.Write	Defines where the data is written. Valid values are: **S.File.WriteAttr.ADDEOF** Data is appended at the end of the file. It is not possible writing or modifying data within the file.
Handle (in)	S.File.Seek S.File.Write S.File.Read S.File.GetLength S.File.Close	An identifier (handle) that identifies the file.
Handle (out)	S.File.Create S.File.ASCreate S.File.Open	An identifier (handle) that identifies the open file. This identifier is returned after a successful call to specified functions and becomes invalid after a call to S.File.Close. If a negative value is returned the file is NOT open - an error occurred.
RetVal	S.File.CreateDir S.File.ASCreateDir S.File.Seek S.File.Write S.File.Read S.File.GetLength S.File.Close S.File.Delete S.File.ASDelete S.File.SeqWrite S.File.ASSeqWrite S.File.StoreString S.File.ASStoreString S.File.SeqRead S.File.ASSeqRead S.File.FormatFS S.File.CompressFS S.File.GetSizeFS S.File.GetDevInfo S.File.GetDevState S.File.FileRename S.File.ASFileRename S.File.GetReleasedSize S.File.EnableAutoCompress	A return value. If smaller than 0 an error occurred during the call. Refer to the error code list. When the function has been successfully executed, the code is 0 is returned. S.File.Read, S.File.ASSeqRead and S.File.SeqRead return the exact number of elements transferred. S.File.GetLength returns the length of the file in bytes.
Template-Text	S.File.StoreString S.File.ASStoreString	A form for Saia PCD® values which are replaces dynamically. Refer to the chapter “Using template texts” for details.
DriveName	S.File.FormatFS S.File.CompressFS S.File.GetSizeFS S.File.GetDevInfo S.File.GetDevState S.File.GetReleasedSize S.File.EnableAutoCompress	Valid device names are: M1_FLASH: On PCD3 extension (M1) M2_FLASH: On PCD3 extension (M2) SL0FLASH: In I/O slot 0 SL1FLASH: In I/O slot 1 SL2FLASH: In I/O slot 2 SL3FLASH: In I/O slot 3

B

Name	Used	Description
BlkSize	S.File.FormatFS S.File.GetDevInfo	Either specifies the internal block size to be used during formatting or the actual internal block size returned from the device information. For PCD7R550 devices, block size is defined from 512 B up to 8KB, each defined step being a multiple of 2 of the previous value. For PCD3R600 devices, block size is defined from 4 KB up to 512 KB, each defined step being a multiple of 2 of the previous value.
BlkNbr	S.File.FormatFS S.File.GetDevInfo	Currently unused Nbr of blocks currently defined for this device
MNoof	S.File.FormatFS S.File.GetDevInfo	Currently unused Maximum number of simultaneous open files
Force	S.File.FormatFS	Setting to1, forces the formatting of the device even if a file system is already present on the device.
TotalSize	S.File.GetSizeFS S.File.GetDevInfo	Size of the file system as it was created (either default size or user provided size)
UsedSize	S.File.GetSizeFS	Currently used size. The used size consists of blocks used either internally by the file system (file list, end of file list, busy list), by the file data, or, in case of PCD7R5xx devices, all blocks that have been used once. To recovers these last block category, a compression has to be performed.
FreeSize	S.File.GetSizeFS	Currently free size where file data can be written or internal file system structure can be written.
CurOpen-Files	S.File.GetDevInfo	Number of files currently open on the given device
Released-Size	S.File.GetReleasedSize	Size which can be recovered by calling the compress system function.
DestName	S.File.FileRename S.File.ASFileRename	The destination filename must be passed as relative name, e.g. NewName.txt. The file name may contain only alphanumeric characters (without SPACE) and ":". A destination filename can not exceed 23 characters, including file extension.
Mode	S.File.EnableAutoCompress	Defines the automatic compression mode. Use either S.File.Compress.AUTO_COMPRESS_ON to enable automatic compression or S.File.Compress.AUTO_COMPRESS_OFF to disable it. *i* Note, that this parameter is reset on each power on of the Saia PCD®.

B

B.2.3 Error codes

In case of success all CFS return in RetVal either 0 (zero) or a positive value. A negative value indicates an error. In this case, please refer to the table below for further information:

B.2.3.1 CSF Internal error codes

RetVal	Value	Explication
-111	CSF_FS_AS_JOBNOTFINISHED	This value is returned if an asynchronous function is called and has not yet been finished. In this case, the same function with identical parameters is to be called again (e.g. in the next program cycle).
-110	CSF_FS_AS_JOBBUSY	This value is returned if a function is called while an asynchronous function has not been finished before.
-109	CSF_FS_LAST_ENTRY_REACHED	This value is returned by the function S.File.GetIndexedFileProp if the given index is bigger than the number of files. When the call returns this value, the returned information does not contain any valid data.
-108	CSF_FS_DENIED	This value is returned if the given GroupAccess / Group parameters are wrong. It also happens when an existing file is accessed and the given group does not correspond to the file group.
-107	CSF_FS_SPACE_MISS	The data read from a file can not be stored in the destination TEXT/DB provided to the function (because it is e.g. not long enough).
-106	CSF_FS_WLEN_TO_BIG	Too many (more than 255) bytes to be written to the file have been specified.
-105	CSF_FS_PLC_WRITE_ERR	Data could not been written to the Saia PCD® media (e.g. because of non existent DB/TEXT).
-104	CSF_FS_DATA_MISS	The DB/TEXT specified is smaller than the amount of the bytes to be written to the file.
-103	CSF_FS_RLEN_TO_BIG	Too many (more than 255) bytes to be read from the file have been specified.
-102	CSF_FS_PLC_READ_ERR	Data could not been read from the Saia PCD® media (e.g. because of non existent DB/TEXT).
-101	CSF_FS_FAILED	This value is returned if the provided file name is invalid (too long, has space or with invalid interpreted characters) or if the provided media is wrong (e.g. a Timer). In this case, please check the CSF parameters and parameter values.

B.2.3.2 File System error codes

RetVal	Value	Explication
-100	FS_WRONG_TYPE	Internal error code, please contact your local Saia Burgess Controls sales office.
-99	FS_DEVICE_NOT_FOUND	The device could not be found, please make sure your flash card is plugged in the right slot.
-98	FS_BAD_PARAMETER	Internal error code, please contact your local Saia Burgess Controls sales office.
-97	FS_INVALID_ARGUMENT	Internal error code, please contact your local Saia Burgess Controls sales office.
-96	FS_FILE_NOT_FOUND	The file could not be found, please make sure the file exists and the path is correct.
-95	FS_INVALID_FILENAME	The specified file name is invalid.
-94	FS_INVALID_GROUP	The specified group is invalid.
-93	FS_INVALID_LEVEL	Internal error code, please contact your local Saia Burgess Controls sales office.
-92	FS_INVALID_ACCTYPE	Internal error code, please contact your local Saia Burgess Controls sales office.
-91	FS_INVALID_DRIVE_NAME	The specified drive name is invalid.
-90	FS_INVALID_DIRECTORY_NAME	The specified directory name is invalid.
-89	FS_FILE_ALREADY_EXIST	This value is returned when trying to create an already existing directory or to rename a file with an existing destination file name.
-88	FS_NOT_ENOUGH_SPACE	Not enough space available on the file system (because it is not compressed or full).
-87	FS_TOO_MANY_OPEN_FILES	No more file can be accessed because too many files are currently open.
-86	FS_FILE_NOT_OPEN	File content can not be accessed because file is not open.
-85	FS_FILE_ALREADY_OPEN	The file can not be opened because it already is open.
-84	FS_INVALID_ACCESS_TYPE	Internal error code, please contact your local Saia Burgess Controls sales office.
-83	FS_INVALID_FILE_TYPE	Internal error code, please contact your local Saia Burgess Controls sales office.
-82	FS_INVALID_WRITE_ATTR	Internal error code, please contact your local Saia Burgess Controls sales office.
-81	FS_INVALID_BUFFER	Internal error code, please contact your local Saia Burgess Controls sales office.
-80	FS_WRITE_ERROR	Internal error code, please contact your local Saia Burgess Controls sales office.
-79	FS_READ_ERROR	Internal error code, please contact your local Saia Burgess Controls sales office.
-78	FS_DAS_ACCESS_REFUSED	Internal error code, please contact your local Saia Burgess Controls sales office (The access to Direct Access fileSystem has been refused).

-77	FS_ACCESS_DENIED	This value is returned after trying to access a file with wrong access attributes, e.g. the file/device is read only and it is accessed with write functions
-76	FS_INV_FILE_DESCR	The file descriptor which has been provided is invalid.
-75	FS_INVALID_USER	Internal error code, please contact your local Saia Burgess Controls sales office.
-74	FS_INVALID_REGFLAGS	Internal error code, please contact your local Saia Burgess Controls sales office.
-73	FS_REG_ENTRY_TABLE_FULL	Internal error code, please contact your local Saia Burgess Controls sales office.
-72	FS_INVALID_REGID	Internal error code, please contact your local Saia Burgess Controls sales office.
-71	FS_FILE_SYSTEM_CHECK_ERROR	Internal error code, please contact your local Saia Burgess Controls sales office.
-70	FS_INV_ENV_NAME	Invalid environment name, please contact your local Saia Burgess Controls sales office.
-69	FS_ENV_NOT_LOADED	Internal error (File system environment not loaded), please contact your local Saia Burgess Controls sales office.
-68	FS_ENV_NAME_ALREADY_EXIST	Internal error (Environment name already exists), please contact your local Saia Burgess Controls sales office.
-67	FS_INVALID_OPERATION	This value is returned after trying to perform a file update on a file/device which does not support this operation. Check file permission / device permission.
-66	FS_INVALID_FLASH_VALUE	Internal error code, please contact your local Saia Burgess Controls sales office. In order to fix this problem, please check FLASH parameters and/or reformat your FLASH device.
-65	FS_FAILED_FLASH_OPERATION	Internal error code, please contact your local Saia Burgess Controls sales office. In order to fix this problem, please check FLASH parameters and/or reformat your FLASH device.
-64	FS_COMPRESSION_ERROR	The device compression failed. This happens when the file system is almost full and the compression had no space on the device to save the new compression parameters. Remove files / directories and perform the compression again.
-63	FS_DEVICE_BUSY	This value is returned while the device is performing an operation (e.g. compression or formatting). Please try executing the function which returned “-63” later.

Remark:

Special take shall be taken when the **FS_DEVICE_BUSY** code is returned. This means that the current device is currently performing an action (e.g. recovering freed blocks) which takes too much time. The function call can NOT wait until this action is finished. The user shall retry later in order to perform its action.

B.2.4 Asynchronous functions

A number of CSF have been implemented to be executed asynchronously. This means the CSF call launches a job which will be executed in background while the user program is executed in parallel. This also implies that the CSF has to be called again with the same parameters until the job is finished. As long as the job is not finished, the CSF call will return a dedicated error code (CSF_FS_AS_JOBNOTFINISHED). If another asynchronous CSF call is issued while the previous job is not finished, the CSF call will return (CSF_FS_AS_JOBBUSY).

These asynchronous CSF have the form S.File.AS<Name>, whereas the same CSF coded synchronously have the form S.File.<Name>.

Two other functions where already provided to be asynchronous. The S.File.FormatFS and S.File.CompressFS which are handled as background job within the file system itself.

An asynchronous job is launched by the CSF call with a defined set of parameters. The same call shall be issued later with the same parameter set. For example:

```
CSF     S.File.Library      ;Access to file system library
        S.File.ASSeqWrite
        FileName                    ;TEXT 4000 = „M1_FLASH:/myfile.txt“
        S.File.GroupId.WEB
        S.File.GroupAccess.WEB
        Content                     ;Source content TEXT 4001
        K 0                         ;Start to read form source at 0
        K 4                         ;Write 4 elements from source
        Result                      ;R 1001
```

shall be called many times with the same parameters (TEXT 4000 for file name, same group identifier, group access, offset and length constants, TEXT 4001 for content and R1001 for result). Giving another register for the result will result in trying to launch another job and the result will be (CSF_FS_AS_JOBBUSY).

A job shall be launched within a COB and its status shall be asked within the same COB.

It is highly not recommended to call file system CSF within XOBs (timing problems).

All asynchronous calls take less than one millisecond to execute.

B.2.5 CSF function execution time

In each of the CSF description, an indication is given concerning its execution time. This execution time is highly dependent on the following factors:

- Ongoing communication (TCP/IP, FTP, HTTP, SBUS Serial, MCx serial communication, USB).
- XOB execution
- Asynchronous CSF background execution
- Other internal tasks execution / internal events
- Device type (PCD7.R5xx or PCD7.R600 – SD cards).

More over, for the same CSF call, the execution time may vary a lot. For example, the S.File.Delete CSF call will take more time if a directory is given as argument (many sub-directories, each of them containing many files) or if a file is given as argument.

For all these reasons, it is impossible to provide precise execution time for the file system CSF.

However, an indication can be provided, which has been measured within the following condition:

- HTTP, FTP and TCP/IP communication are inactive
- No serial communication is active
- In order to start / stop the timing measurements, a PG communication over USB is established, with a very low communication rate (lowest) and only one variable (FLAG) is refreshed.
- The execution time is averaged over many calls (more than 500 calls).
- The device under test has already been used for other test purposes (PCD3.R5xx or PCD3.R600)
- Only one device is tested at a time. SL0FLASH drive has been used.

The given numbers are expressed in milliseconds. One millisecond is the smallest execution time.

These numbers have to be seen as the most probable execution time. It can be used as estimation for the execution time of the according action. Adding communication (HTTP, FTP, Serial) or adding XOB to be executed, will raise the execution time.

Note, that the execution of an asynchronous action will take longer than the execution of its synchronous implementation (because the asynchronous call is executed in parallel to the user program).

B.3 File System SFC Specification

B.3.1 S.File.Create / S.File.ASCreate

S.File.Create		This function creates a file. If the file does not exist, it is created in the specified directory. If the file already exists in the specified directory, it is first deleted and re-created (without data).			
S.File.ASCreate		This function is identical to S.File.Create but is handled asynchronously. Refer to chapter B.2.4 for asynchronous calls description.			
Parameters		4			
1	FileName	X	IN	The file name including the complete path.	
2	GroupId	R K	IN	Group identifier	
3	GroupAccess	R K	IN	Group access parameter	
4	Handle (out)	R	OUT	Returned value (see below)	
Returned Value					
	Success	> 0	File identifier to be used as reference for other file calls.		
	Error	< 0	Refer to chapter B.2.3.		
Timing		Timing Please refer to chapter B.2.5 for comments			
	PCD7R5xx	CSF Error (e.g. file name is too long)			< 1 ms
		FileSystem error - file name too long - directory name too long - directory does not exist - invalid parameter			< 1 ms
		Create successful, file did not exist before call			~ 4 ms
		Create successful, file exist before call (size 256 B)			~100 ms
	PCD3R600	CSF Error (e.g. file name is too long)			< 1 ms
		FileSystem error - file name too long - directory name too long - directory does not exist - invalid parameter			< 1 ms
		Create successful, file did not exist before call			~ 100 ms
		Create successful, file exist before call (size 256 B)			~ 240 ms
Example					

```
CSF       S.File.Library
          S.File.Create (SYNC) or S.File.ASCreate (ASYNC)
          FileName            ;TEXT 4000 = „M1_FLASH:/myfile.txt“
          S.File.GroupID.WEB
          S.File.GroupAccess.WEB
          FileHandle          ;R 1000
```

B

B.3.2 S.File.CreateDir / S.File.ASCreateDir

S.File.CreateDir		This function creates a directory within the specified path.			
S.File.ASCreateDir		This function is identical to S.File.CreateDir but is handled asynchronously. Refer to chapter B.2.4 for asynchronous calls description			
Parameters		4			
1	FileName	X	IN	The directory name including the complete path.	
2	GroupId	R K	IN	Group identifier	
3	GroupAccess	R K	IN	Group access parameter	
4	RetVal	R	OUT	Returned value (see below)	
Returned Value					
	Success	= 0	Directory successfully created.		
	Error	< 0	Refer to chapter B.2.3.		
Timing		Please refer to chapter B.2.5 for comments			
	PCD7R5xx	CSF Error (e.g. directory name is too long)			< 1 ms
		FileSystem error - directory name too long - directory does not exist - invalid parameter - invalid group access			< 1 ms
		Create successful			~ 3 ms
	PCD3R600	CSF Error (e.g. directory name is too long)			< 1 ms
		FileSystem error - directory name too long - directory does not exist - invalid parameter - invalid group access			< 1 ms
		Create successful, level 1 directory			~ 40 ms
Example					

```
CSF       S.File.Library
          S.File.CreateDir (SYNC) or S.File.ASCreateDir (ASYNC)
          FileName           ;TEXT 4000 = „M1_FLASH:/mydir“
          S.File.GroupID.WEB
          S.File.GroupAccess.WEB
          FileResult         ;R 1001
```

B

B.3.3 S.File.Open

S.File.Open		This function allows opening a file, either for reading or for writing. A file handle is returned which is used when accessing the file for the other file access operations. This function can only be called synchronously.			
Parameters		3			
1	FileName	X	IN	The file name including the complete path.	
2	AccessType	R K	IN	Access type parameter	
3	Handle (out)	R	OUT	Returned value (see below)	
Returned Value					
	Success	> 0	File handle to be used as reference for other file calls.		
	Error	< 0	Refer to chapter B.2.3.		
Timing		Please refer to chapter B.2.5 for comments			
	PCD7R5xx	CSF Error (e.g. file name is too long)			< 1 ms
		FileSystem error - file name too long - file does not exist - invalid parameter			< 1 ms
		Open successful (any mode)			< 1 ms
	PCD3R600	CSF Error (e.g. file name is too long)			< 1 ms
		FileSystem error - file name too long - file does not exist - invalid parameter			< 1 ms
		Open successful (any mode)			< 1 ms
Example					

```
CSF       S.File.Library
          S.File.Open
          FileName           ;TEXT 4000 = „M1_FLASH:/myfile.txt“
          S.File.AccessType.RD_WR
          FileHandle         ;R 1000
```

B

B.3.4 S.File.Seek

S.File.Seek		This function allows to navigate within a file by modifying its read or/ and write position, depending on the given parameters. This function can only be called synchronously.		
Parameters		4		
1	Handle (in)	R	IN	A handle to a previously opened file.
2	SeekPos	R K	IN	Relative offset according to current read/write position
3	AccessType	R K	IN	Access type parameter
4	RetVal	R	OUT	Returned value (see below)
Returned Value				
	Success	= 0	Seek operation successfully executed.	
	Error	< 0	Refer to chapter B.2.3.	
Timing		Please refer to chapter B.2.5 for comments		
	PCD7R5xx	FileSystem error - file not open - invalid parameter		< 1 ms
		FileSystem error (e.g. seek after EOF)		< 1 ms
		Seek successful		< 1 ms
		Seek successful (reset pointers)		< 1 ms
	PCD3R600	FileSystem error - file not open - invalid parameter		< 1 ms
		FileSystem error (e.g. seek after EOF)		~ 4 ms
		Seek successful		~ 4 ms
		Seek successful (reset pointers)		< 1 ms
Example				

```
CSF       S.File.Library
          S.File.Seek
          FileHandle          ;R 1000
          K 20
          S.File.AccessType.RD_WR
          CallResult          ;R 1002
```

B

B.3.5 S.File.Close

S.File.Close		This function allows closing a file previously open with the S.File. Open, S.File.Create, S.File.ASCreate operations. This function can only be called synchronously.			
Parameters		2			
1	Handle (in)	R	IN	A handle to a previously opened file.	
2	RetVal	R	OUT	Returned value (see below)	
Returned Value					
	Success	= 0	Close operation successfully executed.		
	Error	< 0	Refer to chapter B.2.3.		
Timing		Please refer to chapter B.2.5 for comments			
	PCD7R5xx	FileSystem error (e.g. file not open)			< 1 ms
		Close successful			< 1 ms
	PCD3R600	FileSystem error (e.g. file not open)			< 1 ms
		Close successful			< 1 ms
Example					

```
CSF       S.File.Library
          S.File.Close
          FileHandle          ;R 1000
          FileResult          ;R 1001
```

B

B.3.6 S.File.Write

S.File.Write		This function allows appending data to a file. The write pointer is updated when writing. This function can only be called synchronously.		
Parameters		6		
1	Handle (in)	R	IN	A handle to a previously opened file.
2	WrAttr	R K	IN	Write attribute parameter
3	Buffer (source)	DB X R	IN	Source buffer from where the data are copied.
4	Offset (source)	R K	IN	The offset from the start of the source buffer.
5	Length	R K	IN	Number of elements to write.
6	RetVal	R	OUT	Returned value (see below)
Returned Value				
	Success	= 0	Write operation successfully executed.	
	Error	< 0	Refer to chapter B.2.3.	
Timing		Please refer to chapter B.2.5 for comments		
	PCD7R5xx	FileSystem error - invalid handle - invalid attribute - register write errors - db write errors - text write errors		< 1 ms
		Write successful: 64 registers		~ 7 ms
		Write successful: 64 DB elements		~ 7 ms
		Write successful: 256 TEXT elements		~ 7 ms
	PCD3R600	FileSystem error - invalid handle - invalid attribute - register write errors - db write errors - text write errors		< 1 ms
		Write successful: 64 registers		~ 40 ms
		Write successful: 64 DB elements		~ 40 ms
		Write successful: 256 TEXT elements		~ 40 ms
Example				

```
CSF       S.File.Library
          S.File.Write
          FileHandle          ;R 1000
          S.File.WrAttr.ADDEOF
          Src                 ;TEXT 4500 = „text string"
          K 0                 ;Start to write to destination at 0
          K 12                ;Write 12 elements from source
          FileResult          ;R 1001
```

B

B.3.7 S.File.Read

S.File.Read		This function allows reading data from a file. The read position is incremented within the open descriptor. This function returns the number of elements (1 or 4 bytes element size) which have been read. If the end of the file is reached while reading, the read position is set at the end of the file and the number of effectively read bytes is returned. This function can only be called synchronously.		
Parameters		5		
1	Handle (in)	R	IN	A handle to a previously opened file.
2	Buffer (destination)	DB X R	IN	The destination to transfer the read data.
3	Offset (destination)	R K	IN	The offset from the start of the destination buffer.
4	Length	R K	IN	Number of elements to read from the file.
5	RetVal	R	OUT	Returned value (see below)
Returned Value				
	Success	≥ 0	Number of effectively read elements.	
	Error	< 0	Refer to chapter B.2.3.	
Timing		Please refer to chapter B.2.5 for comments		
	PCD7R5xx	FileSystem error (e.g. invalid handle)		< 1 ms
		FileSystem error - register read errors - db read errors - text read errors		~ 2 ms
		Read successful: 64 registers		~ 2 ms
		Read successful: 64 DB elements		~ 2 ms
		Read successful: 256 TEXT elements		~ 2 ms
	PCD3R600	FileSystem error (e.g. invalid handle)		< 1 ms
		FileSystem error - register read errors - db read errors - text read errors		~ 4 ms
		Read successful: 64 registers		~ 4 ms
		Read successful: 64 DB elements		~ 4 ms
		Read successful: 256 TEXT elements		~ 4 ms
Example				

```
CSF       S.File.Library
          S.File.Read
          FileHandle          ;R 1000
          Dst       ;DB 4501
          K 0       ;Start to write to destination at 0
          K 10      ;Read 10 DB elements
          Result    ;R 1001
                    ; if successful, Result is 10
```

B

B.3.8 S.File.GetLength

S.File.GetLength		This function allows getting the size of an open file.			
Parameters		2			
1	Handle (in)	R	IN	A handle to a previously open file.	
2	RetVal	R	OUT	Returned value (see below)	
Returned Value					
	Success	>= 0	Actual length of a file in bytes.		
	Error	< 0	Refer to chapter B.2.3.		
Timing		Please refer to chapter B.2.5 for comments			
	PCD7R5xx	FileSystem error (e.g. file not open)			< 1 ms
		Get length successful			~ 1 ms
	PCD3R600	FileSystem error (e.g. file not open)			< 1 ms
		Get length successfu			~ 2 ms
Example					

```
CSF       S.File.Library
          S.File.GetLength
          FileHandle          ;R 1000
          FileLength          ;R 1004
```

B

B.3.9 S.File.Delete / S.File.ASDelete

S.File.Delete		This function allows deleting a file or, if a directory name is given recursively deleting a branch of a file system. This is a synchronous call		
S.File.ASDelete		This function is identical to S.File.Delete but is handled asynchronously. Refer to chapter B.2.4 for asynchronous calls description		
Parameters		3		
1	FileName	X	IN	The file name including the complete path.
2	GroupAccess	R K	IN	Group access parameter.
3	RetVal	R	OUT	Returned value (see below)
Returned Value				
	Success	= 0	Delete operation successfully executed	
	Error	< 0	Refer to chapter B.2.3.	
Timing		Please refer to chapter B.2.5 for comments		
	PCD7R5xx	CSF call error (e.g. name too long)		< 1 ms
		FileSystem error - name too long - file does not exist		< 1 ms
		File delete successful (256 bytes file)		~ 71 ms
		Directory delete successful (3 dir / sub-dir + 2 files)		~ 430 ms
	PCD3R600	CSF call error (e.g. name too long)		< 1 ms
		FileSystem error - name too long - file does not exist		< 1 ms
		File delete successful (256 bytes file)		~ 200 ms
		Directory delete successful (3 dir / sub-dir + 2 files)		~ 1000 ms
Example				

```
CSF       S.File.Library
          S.File.Delete (SYNC) or S.File.ASDelete (ASYNC)
          FileName  ;TEXT 4000 = „M1_FLASH:/myfile.txt”
          S.File.GroupAccess.ALLG
          Result    ;R 1001
```

B

B.3.10 S.File.FileRename / S.File.ASFileRename

S.File.FileRename		This CSF allows renaming a file within a given directory. The source file name can not be a directory name. The destination name is given relative to the source file original path.			
S.File.ASFileR-ename		This function is identical to S.File.FileRename but is handled asyn-chronously. Refer to chapter B.2.4 for asynchronous calls description			
Parameters		4			
1	FileName	X	IN	The file name including the complete path.	
2	DestName	X	IN	The new name of the file	
3	GroupAccess	R K	IN	Group access parameter.	
4	RetVal	R	OUT	Returned value (see below)	
Returned Value					
	Success	= 0	File has been renamed successfully		
	Error	< 0	Refer to chapter B.2.3.		
Timing		Please refer to chapter B.2.5 for comments			
	PCD7R5xx	CSF internal error - Source name too long - destination name too long - Invalid source name - source directory does not exist - absolute name given in destination name - directory information in destination name			< 1 ms
		Rename call successful			~ 5 ms
	PCD3R600	CSF internal error - Source name too long - destination name too long - Invalid source name - source directory does not exist - absolute name given in destination name - directory information in destination name			< 1 ms
		Rename call successful			~ 150 ms
Example					

```
CSF       S.File.Library     ;Access to file system library
          S.File.FileRename (SYNC) or S.File.ASRename (ASYNC)
          SourceName         ;TEXT 3999 =”M1_FLASH:/OrigName.txt”
          DestName  ;TEXT 4000 =“NewName.txt
          S.File.GroupAccess.ALLG      ;Constant
          RetCode            ;R 1001
```

B

B.3.11 S.File.SeqWrite / S.File.ASSeqWrite

S.File.SeqWrite		This function allows writing into a file. Internally to the function, the file is open (created if it does not exist), data are written at the end of the file and file is closed.		
S.File.ASSeqWrite		This function is identical to S.File.SeqWrite but is handled asynchronously. Refer to chapter B.2.4 for asynchronous calls description.		
Parameters		7		
1	FileName	X	IN	The file name including the complete path.
2	GroupId	R K	IN	Group identifier if the file needs to be created.
3	GroupAccess	R K	IN	Group access parameter.
4	Buffer (source)	DB X R	IN	Source buffer from where the data are copied.
5	Offset (source)	R K	IN	The offset from the start of the source buffer.
6	Length	R K	IN	Length of the data to write.
7	RetVal	R	OUT	Returned value (see below)
Returned Value				
	Success	= 0	write operation successfully executed	
	Error	< 0	Refer to chapter B.2.3.	
Timing		Please refer to chapter B.2.5 for comments		
	PCD7R5xx	FileSystem error - invalid attribute - register write errors - db write errors - text write errors		< 1 ms
		Write successful: 64 registers		~ 7 ms
		Write successful: 64 DB elements		~ 7 ms
		Write successful: 256 TEXT elements		~ 7 ms
	PCD3R600	FileSystem error - invalid attribute - register write errors - db write errors - text write errors		< 1 ms
		Write successful: 64 registers		~40 ms
		Write successful: 64 DB elements		~40 ms
		Write successful: 256 TEXT elements		~40 ms
Example				

```
CSF       S.File.Library     ;Access to file system library
          S.File.SeqWrite (SYNC) or S.File.ASSeqWrite (ASYNC)
          FileName           ;TEXT 4000 = „M1_FLASH:/myfile.txt"
          S.File.GroupId.WEB
          S.File.GroupAccess.WEB
          Content            ;Source content TEXT 4000
          K 0                ;Start to read form source at 0
          K 4                ;Write 4 elements from source
          Result             ;R 1001
```

B

B.3.12 S.File.SeqRead / S.File.ASSeqRead

S.File.SeqRead		This function allows reading from a file. Internally to the function, the file is open, data are read and file is closed.		
S.File.ASSeqRead		This function is identical to S.File.SeqRead but is handled asynchronously. Refer to chapter B.2.4 for asynchronous calls description		
Parameters		5		
1	FileName	X	IN	The file name including the complete path.
2	FileOffset	R K	IN	Offset from where to start reading data in the file.
3	Buffer (destination)	DB X R	IN	The destination of the data.
4	Offset (destination)	R	IN	The offset in the destination data.
5	Length	R K	IN	Length of the data to read.
6	RetVal	R	OUT	Returned value (see below)
Returned Value				
	Success	=> 0	Number of effectively read elements.	
	Error	< 0	Refer to chapter B.2.3.	
Timing		Please refer to chapter B.2.5 for comments		
	PCD7R5xx	FileSystem error - register read errors - db read errors - text read errors		~ 2 ms
		Read successful: 64 registers		~ 2 ms
		Read successful: 64 DB elements		~ 2 ms
		Read successful: 256 TEXT elements		~ 2 ms
	PCD3R600	FileSystem error - register read errors - db read errors - text read errors		~ 4 ms
		Read successful: 64 registers		~ 4 ms
		Read successful: 64 DB elements		~ 4 ms
		Read successful: 256 TEXT elements		~ 4 ms
Example				

```
CSF       S.File.Library     ;Access to file system library
          S.File.SeqRead (SYNC) or S.File.ASSeqRead (ASYNC)
          FileName           ;TEXT 4000 = „M1_FLASH:/myfile.txt“
          K 0                ;Start reading from file start
          Content            ;Destination content DB 4000
          K 0                ;Start to write to destination at 0
          K 4                ;Read length 4 DB elements
          Result             ;R 1001
                             ;if successful, Result is 4
```

B

B.3.13 S.File.FormatFS

S.File.FormatFS		This CSF performs the formatting / creation of a file system on a given device. This operation is asynchronous and must be called until the file system has been formatted (return code 0) or an error occurs while formatting (return code < 0). Return code -63 (FS_DEVICE_BUSY) means that the current device is under reformatting.		
Parameters		6		
1	DriveName	X	IN	The drive name
2	BlkSize	R K	IN	Internal block size, expressed in bytes
3	BlkNbr	R K	IN	Set this to 256, currently not used
4	MNoof	R K	IN	Set this to 32, currently not used
5	Force	R K	IN	Setting to 1, forces the formatting of the device even if a file system is already present on the device.
6	RetVal	R	OUT	Returned value (see below)
Returned Value				
	Success	= 0	Formatting of the device is finished.	
	Error	< 0	Refer to chapter B.2.3.	
Timing		Please refer to chapter B.2.5 for comments		
	PCD7R5xx	The CSF execution time is less that 1 ms, but it can take more than 90 seconds to perform the reformatting.		
	PCD3R600	The CSF execution time is less that 1 ms, but it take about 5 seconds to perform the reformatting.		
Example				

```
CSF       S.File.Library     ;Access to file system library
          S.File.FormatFS
          DriveName          ;TEXT 4000 ="M1_FLASH:"
          K 1024
          K 256
          K 32
          K 1                ;Force formatting
          RetCode            ;R 1001
```

B

B.3.14 S.File.CompressFS

This CSF performs the compression / recovery for freed blocks of a file system on a given device. This operation is asynchronous; more than one call is required until the operation is finished.

On the R55Mxx_flash modules, a block is considered either as free (not used yet), as busy (currently used) and freed (has been used at one moment). Freed blocks can not been used until the sector containing the block is erased (all bits set to 1). Only at that moment, a freed block can be re-entered in the free list of blocks.

Internally to the file system, some blocks may be marked as freed, but mainly a block is marked as freed when a file / directory is deleted, all associated blocks being released.

Internally to the file system, the compression (recovery of freed block) is automatically launched when some criteria are met, e.g. the number of freed blocks is 80% of total number of blocks or when the number of freed blocks is bigger than the number of free blocks if this number is less that 1/4th of the total number of blocks. However, this operation can occur at any time and during this operation, the file system is marked as busy. All calls to the file system CSF will then return the FS_DEVICE_BUSY code.

This CSF can be used by the user to force the compression of the device, even if the previous criteria are not met, e.g. if a file is deleted, the user may want to immediately recover all blocks related to that file.

S.File.CompressFS		This CSF Performs the compression / recovery for freed blocks of a file system on a given device. This operation is asynchronous and must be executed to check if the file system has been compressed. Return code -63 (FS_DEVICE_BUSY) means that the current device is under compression.			
Parameters		2			
1	DriveName	X	IN	The drive name including the complete path.	
2	RetVal	R	OUT	Returned value (see below)	
Returned Value					
	Success	= 0	Compression of the device is finished.		
	Error	< 0	Refer to chapter B.2.3.		
Timing		Please refer to chapter B.2.5 for comments			
	PCD7R5xx	The CSF execution time is less that 1 ms, but it can take more than 300 seconds to perform the reformatting.			
	PCD3R600	The CSF execution time is less that 1 ms, but it take about 5 seconds to perform the reformatting.			
Example					

```
CSF       S.File.Library     ;Access to file system library
          S.File.CompressFS
          DriveName          ;TEXT 4000 ="M1_FLASH:"
          RetCode            ;R 1001
```

B.3.15 S.File.EnableAutoCompress

S.File.EnableAutoCompress		This CSF enables to set / reset the auto compression mode implemented in the firmware. By default and at each power ON, the automatic compression is enabled. Disabling the automatic compression avoids the file system to be busy during access operations, e.g. after deleting files or directories. However, without compression, the released blocks are never freed and errors can occur if the file system is full. When disabling the automatic compression, it is highly recommended to perform compression (calling the S.File.CompressFS CSF) on a regular basis or after each deletion of a file. *i* Note, that the EnableAutoCompress is reset to “enabled” on each “power on” of the Saia PCD®.		
Parameters		3		
1	DriveName	X	IN	The drive name including the complete path.
2	Mode	K; R	IN	Defines the automatic compression mode.
3	RetVal	R	OUT	Returned value (see below)
Returned Value				
	Success	= 0	Compression of the device is finished.	
	Error	< 0	Refer to chapter B.2.3.	
Timing		Please refer to chapter B.2.5 for comments		
	PCD7R5xx	The CSF execution time is less that 1 ms		
	PCD3R600	The CSF execution time is less that 1 ms		
Example				

```
CSF       S.File.Library          ; Access to file system library
          S.File.EnableAutoCompress
          DriveName               ; TEXT 4000 ="M1_FLASH:"
          S.File.Compress.AUTO_COMPRESS_ON
                                  ; or AUTO_COMPRESS_OFF
          RetCode                 ; R 1001
```

B

B.3.16 S.File.GetSizeFS

S.File.GetSizeFS		This CSF returns some device information			
Parameters		5			
1	DriveName	X	IN	The drive name	
2	TotalSize	R	OUT	Size of the file system on the given device	
3	UsedSize	R	OUT	Currently used size (including freed blocks).	
4	FreeSize	R	OUT	Currently free size.	
5	RetVal	R	OUT	Returned value (see below)	
Returned Value					
	Success	= 0	Operation successful.		
	Error	< 0	Refer to chapter B.2.3.		
Timing		Please refer to chapter B.2.5 for comments			
	PCD7R5xx	FileSystem error (e.g. Drive name error)			< 1 ms
		Operation successful (R550M04)			~ 2 ms
	PCD3R600	FileSystem error (e.g. Drive name error)			< 1 ms
		Operation successful (512 KB SD card)			~ 20 ms
Example					

```
CSF       S.File.Library     ;Access to file system library
          S.File.GetSizeFS
          DriveName          ;TEXT 4000 = „M1_FLASH:“
          TotalSize          ;R 1000
          UsedSize           ;R 1001
          FreeSize           ;R 1002
          RetCode            ;R 1003
```

B

B.3.17 S.File.GetReleasedSize

S.File.GetReleasedSize		This CSF returns some device information			
Parameters		3			
1	DriveName	X	IN	The drive name	
2	ReleasedSize	R	OUT	Released size on device, but requires a compress in order to recover it.	
3	RetVal	R	OUT	Returned value (see below)	
Returned Value					
	Success	= 0	Operation successful.		
	Error	< 0	Refer to chapter B.2.3.		
Timing		Please refer to chapter B.2.5 for comments			
	PCD7R5xx	FileSystem error (e.g. Drive name error)			< 1 ms
		Operation successful (R550M04)			~ 1 ms
	PCD3R600	FileSystem error (e.g. Drive name error)			< 1 ms
		Operation successful (512 KB SD card)			~ 7 ms
Example					

```
CSF       S.File.Library     ;Access to file system library
          S.File.GetReleasedSize
          DriveName          ;TEXT 4000 = „M1_FLASH:“
          RelSize            ;R 1000
          RetCode            ;R 1001
```

B

B.3.18 S.File.GetDevInfo

S.File.GetDevInfo		This CSF gets some information concerning the given device.			
Parameters		7			
1	DriveName	X	IN	The flash card drive name.	
2	TotalSize	R	OUT	Size of the file system as it was created.	
3	BlkSize	R	OUT	Block size of the current file system.	
4	BlkNbr	R	OUT	Number of blocks.	
5	MNoof	R	OUT	Maximum number of open files.	
6	CurOpenFiles	R	OUT	Current number of open files.	
7	RetVal	R	OUT	Returned value (see below)	
Returned Value					
	Success	= 0	Operation successful.		
	Error	< 0	Refer to chapter B.2.3.		
Timing		Please refer to chapter B.2.5 for comments			
	PCD7R5xx	FileSystem error (e.g. Drive name error)			< 1 ms
		Operation successful			< 1 ms
	PCD3R600	FileSystem error (e.g. Drive name error)			< 1 ms
		Operation successful			< 1 ms
Example					

```
CSF       S.File.Library     ;Access to file system library
          S.File.GetDevInfo
          DriveName ;TEXT 4000 = „M1_FLASH:“
          Size      ;R 1000
          BlockSize ;R 1001
          BlockNbr  ;R 1002
          MaxOpenFiles       ;R 1003
          CurOpenFiles       ;R 1004
          RetCode   ;R 1005
```

B

B.3.19 S.File.GetDevState

S.File.GetDevState		Get the current status of the device.		
Parameters		2		
1	DriveName	X	IN	The flash card drive name.
2	RetVal	R	OUT	Returned value (see below)
Returned Value				
	Status	= 0	FS_FILE_SYSTEM_OK: File system is OK and accessible	
	Status	= -63	FS_DEVICE_BUSY: File system is currently busy with internal jobs, e.g. compression	
	Status	= -71	FS_FILE_SYSTEM_CHECK_ERROR: At start-up, the file system could not be re-created due to internal errors, but the correct device has been found.	
	Status	= -99	FS_DEVICE_NOT_FOUND: No correct devices found at that location.	
Timing		Please refer to chapter B.2.5 for comments		
	PCD7R5xx	FileSystem error (e.g. Drive name error)		< 1 ms
		Operation successful		< 1 ms
	PCD3R600	FileSystem error (e.g. Drive name error)		< 1 ms
		Operation successful		< 1 ms
Example				

```
CSF       S.File.Library     ;Access to file system library
          S.File.GetDevState
          DriveName ;TEXT 4000 = "M1_FLASH:"
          RetCode   ;R 1000
```

B

B.3.20 S.File.GetIndexedFileProp

S.File.GetIndexed-FileProp		This CSF allows to retrieving files within a named directory. Using the index argument, it is possible to through the list of files present on a device. When no files are available anymore, a dedicated error code is returned.		
Parameters		8		
1	FileName	X	IN	Name of a directory (See below)
2	GroupAccess	K	IN	Allow to discard files which are not member of the given group access.
3	Index	R	IN	Defines which file has to be taken from the directory.
4	FileName	X	OUT	Return the indexed file from the directory, containing the full path name information.
5	FileType	R	OUT	Returns the type of the file. Value of 3 means it is a file, value of 1 means it is a directory.
6	AccessType	R	OUT	Returns the access type defined for the file.
7	GroupId	R	OUT	Returns the group identifier of the file.
8	RetVal	R	OUT	Returned value (see below)
Returned Value				
	Success	= 0	Operation successful.	
	Error	< 0	Refer to chapter B.2.3.	
Timing		Please refer to chapter B.2.5 for comments		
	PCD7R5xx	The CSF execution time is less that 1 ms		
	PCD3R600	The CSF execution time is less that 1 ms		
Example				

```
CSF       S.File.Library     ;Access to file system library
          S.File.GetIndexedFileProp
          Filename  ;TEXT 4000 = „M1_FLASH:/WEBPAGES/“
          S.File.GroupAccess.ALLG
          Index     ;R 1000
          Fullname  ;TEXT 4001 = “MI_FLASH:/WEBPAGES/P.HTML”
          FType     ;R 1001
          AccType   ;R 1002
          FroupId   ;R 1003
          RetCode   ;R 1004
```

Extended features / remarks:

- It is possible to list only files with a defined extension (e.g. .html in this case) by giving M1_FLASH:/WEBPAGES/.HTML as input name.
- It is possible to list only files containing a given string (e.g. Box in this case) by giving M1_FLASH:/WEBPAGES/BOX as input name.
- If the input name is M1_FLASH:/WEBPAGES/ all files from that directory will be listed, whereas if the input name is M1_FLASH:/WEBPAGES all files containing the string WEBPAGES from the root directory will be listed.
- Starting from index 0, the first file will be returned compliant with the input parameters (input filename and group access. By incrementing the index, the next file will be returned. When no file is left, the return code is CSF_FS_LAST_ENTRY_REACHED (-109).
- The order of files returned by this function is random, it is neither alphabetic, nor time related nor files or directories.

B.3.21 S.File.StoreString / S.File.ASStorString

S.File.StoreString		This function allows writing a template text to a file. - If the file not yet exists, it creates, opens, appends the text and closes the file. - If the file exists already, it appends the text and closes the file. The text can contain template text elements described in the following chapter.			
S.File.ASStor-eString		This function is identical to S.File.StoreString but is handled asynchronously. Refer to chapter B.2.4 for asynchronous calls description			
Parameters		5			
1	FileName	X	IN	The file name including the complete path, this text is also interpreted.	
2	TemplateText	X	IN	A template text with tags which will result in a formatted text, maximum resolved length is 256 bytes. Refer to chapter 4 for some information concerning the available templates.	
3	GroupId	R K	IN	Defines to which group the create file / directory belongs.	
4	GroupAc-cess	R K	IN	This parameter allows to access files / directories belonging to one of the given groups. Any combi-nation of the above defined group can be defined.	
5	RetVal	R	OUT	Returned value (see below)	
Returned Value					
	Success	= 0	Operation successful.		
	Error	< 0	Refer to chapter B.2.3		
Timing		Please refer to chapter B.2.5 for comments			
	PCD7R5xx	FileSystem error (e.g. invalid attribute), but parsing is done for a 256 bytes text.			~ 2 ms
		FileSystem error (e.g. text write errors), but pars-ing is done with many interpreted text IDs, e.g. $D,$Lxxxx.			~ 7 ms
		Write successful: 256 TEXT elements, but no tags are present in source text.			~ 7 ms
	PCD3R600	FileSystem error (e.g. invalid attribute), but parsing is done for a 256 bytes text.			~ 2 ms
		FileSystem error (e.g. text write errors), but pars-ing is done with many interpreted text IDs, e.g. $D,$Lxxxx.			~ 7 ms
		Write successful: 256 TEXT elements, but no tags are present in source text.			~ 40 ms
Example		`This example creates a comma separated values file (csv) with Saia PCD® values. Every 60 second and raising edge of input 1 triggers the StoreString function. No error handling is done.`			

B

```
CsvFile          = „M2_FLASH:/logfile.csv“   (Text 4000)
CsvContent       = „@L“,CsvCurrentText,“;$d$H;$“,ADCH0.T,“;$“,ADCH1.T,“;
$“,ADCH2.T,“;$“,ADCH3.T,“ <CR><LF>“        (Text 4002)
CsvEvent1        = „60 second event“        (Text 4010)
CsvEvent2        = „Input 1 event“          (Text 4011)
CsvCurrentText(R), CsvResult(R), LogTrigger(T), Input0(I 0), Input1(I 1)
```

```
$INCLUDE „FileSystem.inc“
COB   0
      0
      LD       CsvCurrentText
               4010
      STL      LogTrigger          ;and timer = 0
      LD       LogTrigger          ;then load timer
               T#60S               ;with 60 sec
      CSF   H  S.File.Library      ;Access to file system
               S.File.StoreString
               CsvFile             ;Filename text
               CsvContent          ;Content template text
               S.File.GroupId.WEB
               S.File.GroupAccess.WEB
               CsvResult
      STH      Input1
      DYN   F  0
      LD       CsvCurrentText
               4011
      CSF   H  S.File.Library      ;Access to file system
               S.File.StoreString
               CsvFile ;Filename text
               CsvContent   ;Content template text
               S.File.GroupId.WEB
               S.File.GroupAccess.WEB
               CsvResult
ECOB
```

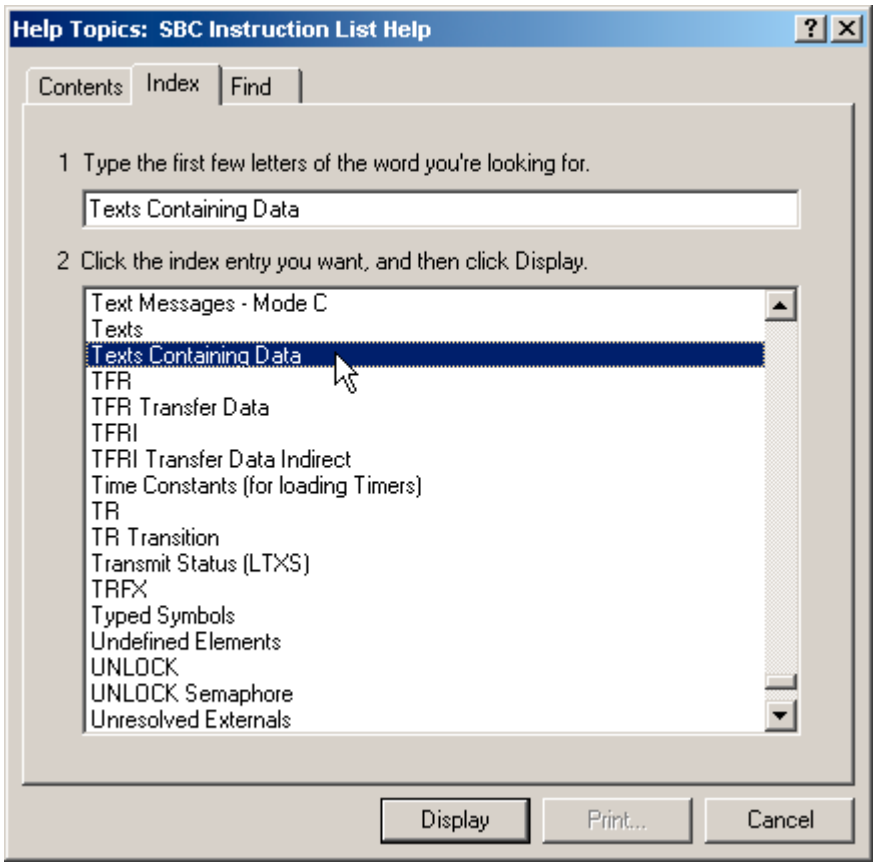
B

B.4 Template texts

Texts can also contain data values such as the clock, the status of an input, the contents of a register etc. There are two characters which have a special meaning for the Saia PCD®: $ and @.

If a text contains e.g. the expression $R0010. this expression will be replaced with the current content of Register 10, while writing into a file.

This allows writing a line into a file that contains the current date, time, the current content of a Saia PCD® media and other information. For further information, regarding template texts and their format, open the “Instruction List Help” of the Saia PG5® Instruction List Editor (S-Edit) Help Menu. Click the button “Index” in the Online Help of S-Edit and type “text”. Select (Double-Click) “texts contaiting data” from the appearing selection of topics, as shown in the screenshot.



B

C Appendix

C.1 Icons

	In manuals, this symbol refers the reader to further information in this manual or other manuals or technical information documents. As a rule there is no direct link to such documents.
	This symbol warns the reader of the risk to components from electrostatic discharges caused by touch. Reco mmendation : at least touch the Minus of the system (cabinet of PGU connector) before coming in contact with the electronic parts. Better is to use a grounding wrist strap with its cable attached to the Minus of the system.
	This sign accompanies instructions that must always be followed.
	Explanations beside this sign are valid only for the Saia PCD® Classic series
	Explanations beside this sign are valid only for the Saia PCD® xx7 series.

C

C.2 Contact

Saia-Burgess Controls AG
Bahnhofstrasse 18
3280 Murten
Switzerland

Phone ..... +41 26 580 30 00
Fax..... +41 26 580 34 99

Email support: ..... support@saia-pcd.com
Supportsite: ..... www.sbc-support.com
SBC site: ..... www.saia-pcd.com
International Represetatives &
SBC Sales Companies: ..... www.saia-pcd.com/contact

Postal address for returns from customers of the Swiss Sales office

Saia-Burgess Controls AG
Service Après-Vente
Bahnhofstrasse 18
3280 Murten
Switzerland

C